Desenvolupament d’una aplicació de filtratge web

José Juan Sanz
Grau en Enginyeria Informàtica

Maria Isabel March Hermo

19/06/2018

FITXA DEL TREBALL FINAL

Títol del treball:	*Desenvolupament d'una aplicació de filtratge web*
Nom de l'autor:	*José Juan Sanz*
Nom del consultor:	*Maria Isabel March Hermo*
Data de lliurament (mm/aaaa):	*06/2018*
Àrea del Treball Final:	*Xarxes de computadors*
Titulació:	*Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

Avui dia qualsevol persona té a l'abast de la mà en gairebé qualsevol moment un dispositiu amb connexió a Internet per accedir a qualsevol contingut. Ja siguem adults o infants, estiguem a casa, a la feina o a l'escola, podem accedir a recursos de la xarxa amb molta facilitat.

El que sembla un clar avantatge pot esdevenir en inconvenient en algunes situacions. Per exemple, aquesta facilitat d'accés pot provocar una distracció continua durant hores de feina o estudi, produint una reducció de la nostra productivitat. Un altre clar inconvenient és que els infants tenen la mateixa facilitat per accedir a qualsevol contingut, també als que no son convenients per a la seva edat.

Per a resoldre aquesta problemàtica existeixen les aplicacions de filtratge web. Aquestes aplicacions tenen com a funcionalitat principal poder filtrar la navegació a Internet que fan els usuaris i bloquejar el seu accés total o parcial.

Aquest Treball de Fi de Grau té la finalitat de desenvolupar una aplicació de filtratge web que permeti una configuració personalitzada dels paràmetres de filtratge. Amb la realització d'aquest treball vull aprofundir en el coneixement de com funcionen aquest tipus d'aplicacions i aprofitar aquest coneixement per a desenvolupar una aplicació que sigui completament funcional i accessible per a qualsevol usuari.

Abstract (in English, 250 words or less):

Nowadays, almost anybody has access to the Internet and is able to consume its contents. Adults or children, being home, at work or at school, we can access the Internet resources in an easy way.

What it seems to be a clear advantage it could become a downside in some situations. For example, that easy access could cause a continuous distraction during work or study time, which could cause low productivity. Another disadvantage is that children are able to consume content of any kind, also content which is not appropriate for them.

Web filtering applications were created to solve those problems. This kind of applications has as the main functionality the ability to filter the Internet navigation of the users and block their access.

This project has the purpose of developing a web filtering application which allows customizing the filtering parameters. With this project, I want to look into the knowledge of how this kind of applications work and use this knowledge to develop a real application fully functional and available to any user.

Paraules clau (entre 4 i 8):

Web
Filtratge
Bloqueig
HTTP
Aplicació
Navegació

Índex

Llista de figures

1. Introducció

En aquest apartat inicial es presenta el context, objectius, enfocament i resultat del projecte, així com la planificació inicial i el que ens podrem trobar en la resta d'apartats de la memòria.

1.1 Context i justificació del Treball

Avui dia qualsevol persona té a l'abast de la mà en gairebé qualsevol moment un dispositiu amb connexió a Internet per accedir a qualsevol contingut. Ja siguem adults o infants, estiguem a casa, a la feina o a l'escola, podem accedir a recursos de la xarxa amb molta facilitat.

El que sembla un clar avantatge pot esdevenir en inconvenient en algunes situacions. Per exemple, aquesta facilitat d'accés pot provocar una distracció continua durant hores de feina o estudi, produint una reducció de la nostra productivitat. Un altre clar inconvenient és que els infants tenen la mateixa facilitat per accedir a qualsevol contingut, també als que no son convenients per a la seva edat.

Per a resoldre aquesta problemàtica existeixen les aplicacions de filtratge web. Aquestes aplicacions tenen com a funcionalitat principal poder filtrar la navegació a Internet que fan els usuaris i bloquejar el seu accés total o parcial.

Aquest Treball de Fi de Grau té la finalitat de desenvolupar una aplicació de filtratge web que permeti una configuració personalitzada dels paràmetres de filtratge. Amb la realització d'aquest treball vull aprofundir en el coneixement de com funcionen aquest tipus d'aplicacions i aprofitar aquest coneixement per a desenvolupar una aplicació que sigui completament funcional i accessible per a qualsevol usuari.

1.2 Objectius del Treball

Els objectius a aconseguir amb el desenvolupament d'aquest treball són els següents:

- Desenvolupar una aplicació que permeti filtrar la navegació web que fan els usuaris i, segons uns paràmetres establerts, bloquejar les pàgines que compleixin unes condicions determinades.

- Permetre modificar aquests paràmetres en que es basa el bloqueig en qualsevol moment i des de qualsevol lloc, i que la seva aplicació sigui instantània.

- Utilitzar el filtre per a guardar un registre de l'activitat de navegació dels usuaris. Aquest registre es podrà consultar també en qualsevol moment i des de qualsevol lloc.

- Fer accessible aquesta aplicació a qualsevol usuari.

1.3 Enfocament i mètode seguit

El primer pas va ser documentar-me sobre el protocol HTTP i les tecnologies de filtratge web, i revisar quines aplicacions d'aquest tipus existeixen actualment, i com funcionen. Això em va ajudar a definir quins tipus d'aplicacions hi ha al mercat, quin format tenen, i estudiar els seus avantatges i inconvenients.

A continuació vaig definir i limitar les funcionalitats que havia de tenir l'aplicació, i un cop decidit vaig determinar quin tipus d'aplicació implementaria, una extensió de navegador juntament amb una API REST que serviria per a centralitzar les dades dels usuaris i el registre de navegació.

En aquest punt vaig fer una petita prova implementant una extensió de navegador de prova per a validar que amb aquest tipus d'aplicació podia aconseguir les funcionalitats que havia definit prèviament.

Un cop feta aquesta validació, vaig iniciar l'etapa d'anàlisi i disseny, definint quins tipus d'usuaris hi haurien, el model de dades, casos d'ús, el diagrama de flux de l'aplicació i els *wireframes* de les diferents pantalles.

Seguidament vaig fer el gruix de la implementació de l'extensió i de la API REST. Vaig fer la instal·lació i configuració de la API REST a un servidor d'accés públic i vaig empaquetar l'extensió per a permetre la seva instal·lació a qualsevol navegador Firefox.

Per últim, vaig redactar el manual d'ús de l'aplicació i vaig fer les proves per validar el correcte funcionament de l'aplicació.

1.4 Planificació del Treball

La planificació inicial del treball va ser la següent, encara que per manca de temps a les primeres setmanes no vaig poder seguir-la com m'hagués agradat. Això va provocar que al tram final hagués de fer un esforç extra per tenir-ho tot enllestit.

Tasca 1 - Del 5 de març al 25 de març:

Buscar informació i documentació sobre possibles tecnologies i aplicacions a utilitzar per a la implementació de l'aplicació de filtratge de pàgines web.
Destacar punts forts i febles sobre la informació obtinguda i decidir les tecnologies a utilitzar al projecte d'acord amb aquests punts.
Fer una especificació detallada de les funcionalitats que tindrà l'aplicació, de major a menor prioritat:

- Què filtrarà i com es farà el filtratge.
- Com es realitzarà el control horari.
- Quines mètriques i estadístiques es mostraran.

L'objectiu d'aquesta etapa és escollir la tecnologia amb la que es desenvoluparà l'aplicació i tenir definides les funcionalitats que tindrà.

Tasca 2 - Del 25 de març al 15 d'abril:

Definir el model de dades i els casos d'ús de l'aplicació.
Fer el disseny de la interfície d'usuari de l'aplicació, obtenint uns mockups inicials.
Iniciar el desenvolupament de l'aplicació, que contingui un esquelet inicial de la mateixa.
L'objectiu a aconseguir és tenir una primera versió que es pugui instal·lar al navegador, que tingui un esquelet que permeti canviar entre la part de configuració de llistat de webs permeses i definició d'horari, i la part de visualització d'estadístiques. En aquesta primera versió aquestes dues parts estaran buides.

Tasca 3 - Del 16 d'abril al 6 de maig:

Implementar la part de l'aplicació que s'encarregarà de capturar la navegació de l'usuari i permetrà que l'usuari pugui navegar o no d'acord amb el que hi hagi definit a la configuració de webs permeses i horaris definits. En cas que no es permeti la navegació, provocar que es bloquegi el navegador i es mostri un missatge informatiu a l'usuari amb el motiu del bloqueig.
Desenvolupar la part visual on l'usuari podrà guardar la configuració.
L'objectiu final d'aquesta tasca és permetre que la part de l'aplicació des d'on es pot configurar el filtratge web sigui operativa, i que el filtratge funcioni correctament.

Tasca 4 - Del 7 de maig al 27 de maig:

Desenvolupar la part encarregada d'enregistrar tota la informació de navegació dels usuaris. La informació a enregistrar serà la que s'haurà definit a la tasca 1.
Fer el desenvolupament de la part del dashboard de visualització d'aquesta informació.
En aquesta tasca l'objectiu serà tenir operativa la part de l'aplicació on es mostren les mètriques de navegació, i amb això ja tindrem una primera versió de l'extensió amb totes les funcionalitats operatives.

Tasca 5 - Del 28 de maig al 10 de juny:

Realitzar proves de l'aplicació per comprovar el seu correcte funcionament i solucionar possibles errors trobats.
Finalitzar el document de la memòria del treball.
L'objectiu d'aquesta etapa és tenir una versió de l'aplicació totalment funcional i sense errors, i tenir-ho tot documentat a la memòria de treball.

Tasca 6 - De l'11 de juny al 17 de juny:

Preparar la presentació del TFG.

1.5 Breu sumari de productes obtinguts

Els productes obtinguts han estat:

- Una extensió per al navegador Firefox que permet:
    - Establir una configuració de filtratge per a bloquejar l'accés a pàgines web segons els següents paràmetres:
        - La web conté contingut per a adult.
        - La URL de la pàgina web.
        - El servidor web utilitzat per servir la pàgina web.
        - La web conté paraules concretes.
    - Consultar un registre amb la navegació realitzada pels usuaris, amb la informació de quines pàgines han estat blocades i el motiu del bloqueig.
    - Filtrar la navegació web dels usuaris i bloquejar l'accés a algunes pàgines web segons la configuració guardada prèviament.
- Una API REST que utilitza l'extensió per a gestionar la informació dels usuaris i la seva configuració, i per a enregistrar la seva activitat de navegació.
- Un document amb la memòria del projecte realitzat.

1.6 Breu descripció dels altres capítols de la memòria

- Capítol 2, conceptes tècnics: es parla del protocol HTTP, de la teoria de les aplicacions de filtratge web, i s'estudien aplicacions de filtratge web ja existents al mercat.

- Capítol 3, especificació: es defineixen les funcionalitats que ha de tenir l'aplicació de filtratge web.

- Capítol 4, anàlisi i disseny: s'exposa el procés d'anàlisi i disseny de l'aplicació, justificant les decisions que s'han pres. Més concretament, es justifica l'elecció del format d'aplicació a implementar (una extensió de navegador), es defineixen els rols d'usuari, el model de dades, els casos d'ús, el diagrama de flux de l'aplicació i els *wireframes* amb el disseny de les pantalles.

- Capítol 5, implementació: en primer lloc es presenta la documentació de les llibreries utilitzades per a fer la intercepció de les peticions HTTP i poder fer el filtratge. A continuació es defineix l'arquitectura de l'aplicació, i s'explica la implementació dels components: la base de dades, l'API REST i l'extensió de navegador.

- Capítol 6, instal·lació i configuració de l'entorn de producció: es detalla el procés seguit per a instal·lar i configurar el servidor de producció que donarà servei a l'aplicació, i el procés per a fer el paquet instal·lable de l'extensió del navegador.

- Capítol 7, manual d'ús: conté el manual d'ús de l'aplicació.

- Capítol 8, proves de l'aplicació: s'exposa les proves que s'han realitzat per a validar el correcte funcionament de l'aplicació.

2. Conceptes tècnics

En aquest apartat s'exposarà com funcionen les aplicacions de filtratge web i les tecnologies i protocols que fan servir.

2.1. Protocol HTTP

Per a realitzar el filtratge web haurem d'analitzar els missatges HTTP que s'envien entre el client i el servidor.

El protocol de transferència d'hipertext (HTTP) és el protocol de comunicació que permet les transferències d'informació a Internet.

HTTP va ser desenvolupat pel World Wide Web Consortium (WWWC) i la Internet Engineering Task Force (IETF) en una col·laboració que va culminar el 1999 amb la publicació d'una sèrie de RFC, incloent-hi el RFC 2616 que especifica la versió 1.1.

HTTP defineix la sintaxi i la semàntica que utilitzen els elements de software de l'arquitectura web per a comunicar-se. HTTP és un protocol sense estat, per la qual cosa cada connexió és independent i no es guarda informació sobre connexions anteriors.

HTTP és un protocol orientat a transaccions i segueix l'esquema petició-resposta entre un client i un servidor. El client (anomenat 'agent d'usuari') realitza una petició enviant un missatge amb un format concret al servidor, i el servidor web retorna un missatge de resposta.

Els missatges HTTP són en text pla, i tenen la següent estructura:

- Línia inicial amb un contingut diferent segons si és una petició o una resposta:
  - Si és una petició, la línia inicial conté l'acció que es demana al servidor (el mètode de la petició) juntament amb la URL del recurs i la versió HTTP que suporta el client.
  - Si és una resposta, la línia inicial conté la versió del HTTP utilitzat seguit del codi de resposta i de la frase associada al retorn.

- Les capçaleres del missatge, que finalitzen amb una línia en blanc.

- El cos del missatge, que és opcional, i que conté les dades que s'intercanvien el client i el servidor.

A continuació es mostra un exemple dels missatges intercanviats per demanar la web http://developer.mozilla.org:

- El client fa una petició a la URL amb el mètode GET, que indica que volem demanar un recurs al servidor. S'inclou el *path* del recurs que volem al servidor i la versió del protocol que utilitza el client. També s'envien diferents capçaleres, com el host de la URL que demanem.

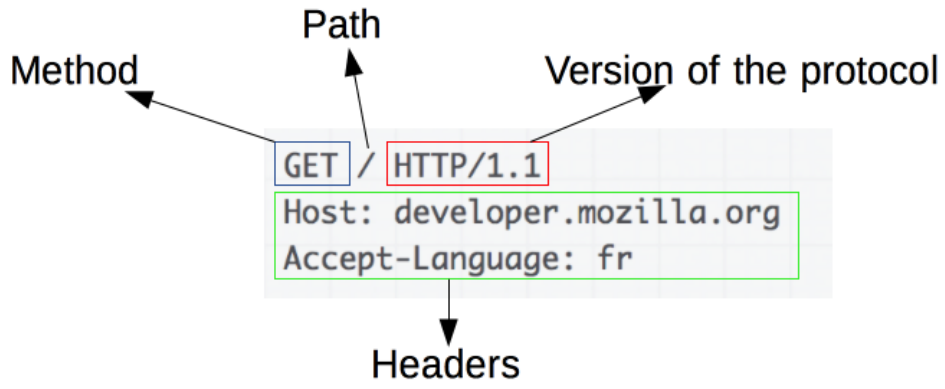


Figura 1: Exemple de missatge de petició HTTP

- El servidor ens respon amb un missatge que inclou el *status* de la resposta (que indica si hi ha hagut algun error o no), la versió del protocol HTTP, les capçaleres de la resposta, i el *body* de la resposta (contingut del recurs que hem demanat), que és opcional.

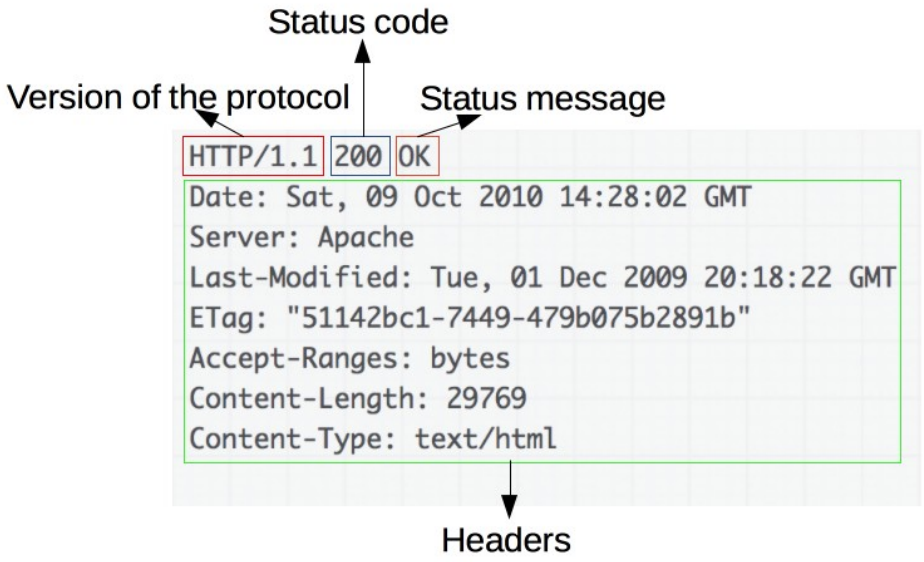


Figura 2: Exemple de missatge de resposta HTTP

2.2. Aplicacions de filtratge web

Una aplicació de filtratge web és una aplicació que és capaç de monitorar una petició web i determinar si la pàgina web de la petició es mostrarà o no. Aquesta decisió es farà d'acord amb un conjunt de regles configurades prèviament per la companyia responsable de l'aplicació o per l'usuari de l'aplicació.

L'aplicació de filtratge web també pot encarregar-se d'emmagatzemar la informació de quines webs s'ha demanat accés, quines d'aquestes webs s'han bloquejat i per quin motiu, quins usuaris han intentat accedir a les diferents webs i en quin moment, etc.

Les aplicacions de filtratge web poden treballar de dues maneres:

1. Realitzant el filtratge basant-se en la petició:

Aquest tipus de filtratge consisteix a inspeccionar la petició que realitza l'usuari per determinar si la petició es bloquejarà o no. Aquesta és una manera molt àgil i lleugera de realitzar el filtratge, ja que en una petició hi ha molta menys informació a analitzar, però per aquest mateix motiu és també molt limitada. Es podria fer servir informació de connexions anteriors o d'altres usuaris per a decidir sobre el filtratge però això no és fiable, ja que el contingut d'una pàgina web pot canviar amb el temps.

Aquest és un mètode que es pot fer servir per a bloquejar peticions basant-se en 'blacklists' d'urls que s'han identificat prèviament i que ens evita haver d'analitzar la resposta de la petició, però no és suficient si volem implementar una aplicació de filtratge completa.

2. Realitzant el filtratge web basant-se en la resposta de la petició:

Aquest tipus de filtratge consisteix a inspeccionar els paquets de la resposta que ens retorna el servidor. És un mètode més pesat que l'anterior perquè s'han d'analitzar les capçaleres de la resposta i el contingut del cos de la resposta, però d'aquesta manera també ens podem assegurar que la web que es mostrarà a l'usuari compleix tots els requisits definits a la configuració de l'aplicació de filtratge web.

2.3. Investigació d'aplicacions de filtratge web existents

Algunes de les aplicacions de filtratge web que ja existeixen al mercat són:

- Qustodio:

  És una aplicació de control parental que es ven com una eina de supervisió per a l'activitat online dels nens. És una aplicació compatible amb Windows, Mac, Android, iOS, Kindle i Nook, que s'ha d'instal·lar en tots els dispositius d'aquest tipus a on volem fer la supervisió. Ofereix un panel de control accessible per als pares (en forma d'aplicació o de pàgina web) per a poder gestionar la configuració dels filtres, establir un horaris d'ús, veure mètriques de la navegació realitzada pels fills, etc.

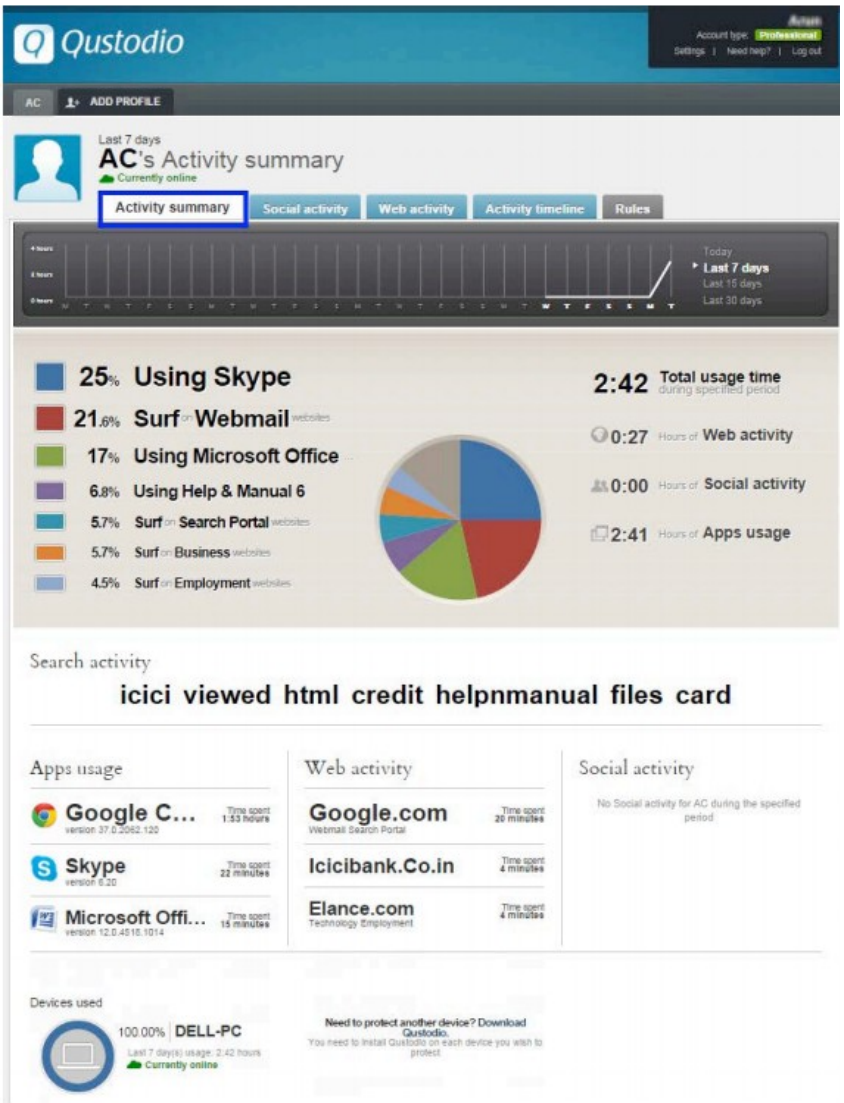


Figura 3: Imatge del panell d'administració de Qustodio amb la informació de l'activitat de navegació

- Block Site - Website Blocker for Chrome:

  És una extensió de navegador que permet bloquejar l'accés a les pàgines web que haguem definit.

  També insereix un botó al navegador per a poder activar i desactivar fàcilment tant el bloqueig general com el mode treball. A més a més, des d'aquí també es pot activar un bloqueig de pàgines destinades a adults.

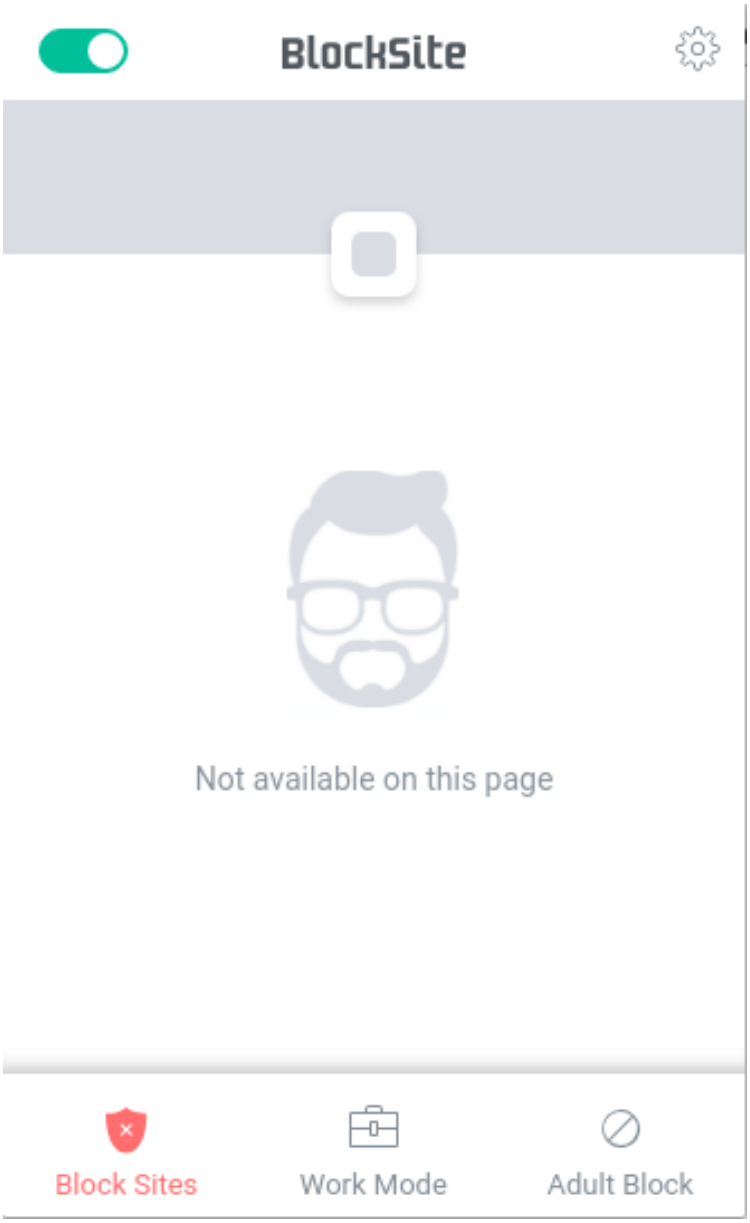


Figura 4: Popup que mostra BlockSite al fer clic al botó del navegador

Té una pàgina d'administració amb tres opcions de bloqueig:

- Block Sites: per a afegir les direccions de les webs que es volen bloquejar permanentment.

- Work Mode: per a afegir les direccions de les webs que es volen bloquejar quan el mode *work mode* està activat. Aquest mode serveix per a evitar distraccions a l'usuari quan està treballant.

- Block by Words: per a afegir les paraules que volem bloquejar. Si alguna web de les que visitem conté alguna de les paraules afegides, es bloqueja l'accés.

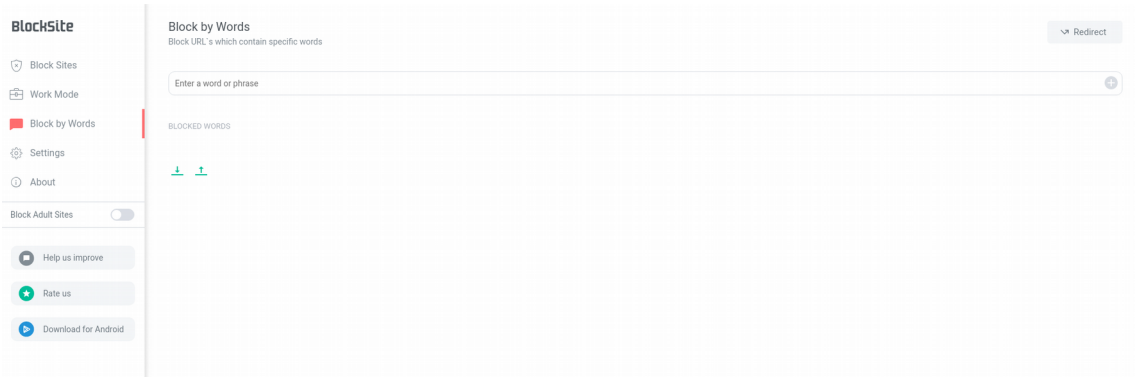


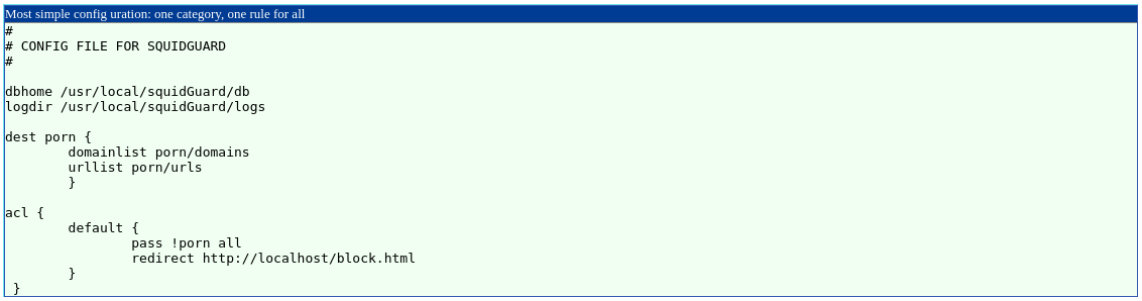
Figura 5: Pàgina d'administració de BlockSite

- SquidGuard:

  SquidGuard és un plugin per a Squid (un servidor proxy per a web amb catxe), que compara les peticions dels usuaris amb un conjunt de blacklists. Aquest conjunt de blacklists es pot configurable. És a dir, aquest software funciona com un servidor proxy que intercepta les peticions dels usuaris i revisa si la web a la que s'està accedint pertany a alguna de les blacklists configurades i si és el cas fa una redirecció a una pàgina de bloqueig per a evitar que l'usuari pugui accedir.

  Com que és un servidor proxy, aquest tipus de filtratge és vàlid per a qualsevol dispositiu, però el dispositiu s'ha de configurar per a que les peticions http passin per el servidor proxy, o bé s'ha de modificar la configuració de la xarxa local a on està el dispositiu que es vol filtrar.

  La configuració de SquidGuard es pot canviar modificant el seu fitxer de configuració, indicant quines categories de webs es volen filtrar escollint les blacklists disponibles.


Most simple config uration: one category, one rule for all

```
#
# CONFIG FILE FOR SQUIDGUARD
#

dbhome /usr/local/squidGuard/db
logdir /usr/local/squidGuard/logs

dest porn {
        domainlist porn/domains
        urllist porn/urls
        }

acl {
        default {
                pass !porn all
                redirect http://localhost/block.html
        }
 }
```

Figura 6: Exemple de fitxer de configuració de SquidGuard

3. Especificació

L'aplicació de filtratge tindrà dos funcionalitats principals, que es detallen a continuació:

1. Bloquejar l'accés a determinades pàgines web: el bloqueig es farà segons uns paràmetres definits per l'usuari, que es podran modificar en qualsevol moment. Aquests paràmetres de filtratge seran els següents:

    a) Habilitar o deshabilitar el filtratge web: es podrà activar o desactivar l'aplicació de forma general.
    b) Llistat de dominis a bloquejar: es podrà configurar un llistat amb els dominis a bloquejar per part de l'aplicació. Aquest serà un filtratge del tipus filtratge basant-se en la petició, ja que analitzarà la request de la petició http per veure el domini de la petició. Si el domini coincideix amb algun dels dominis definits a la configuració, la petició es bloquejarà. En aquest tipus de filtratge la petició http no s'arriba a realitzar.
    c) Llistat de servidors a bloquejar: es podrà configurar un llistat amb els tipus de servidors que voldrem bloquejar. Aquest ja és un filtratge del tipus filtratge basant-se en la resposta, ja que hem d'observar el header *Server* que envia el servidor en la resposta de la petició HTTP. Si el servidor ens envia aquest header i coincideix amb algun dels termes que tenim al llistat de servidors la petició es bloquejarà. Com que podem obtenir aquesta informació sense haver d'esperar al contingut del cos de la resposta de la petició HTTP, aquest bloqueig serà més lleuger i àgil que els bloquejos que han d'analitzar el cos de la resposta.
    d) Llistat de paraules a bloquejar: a la configuració també podrem construir un llistat amb paraules que volem bloquejar. L'aplicació bloquejarà la pàgina web en cas que hi hagi alguna de les paraules del llistat en el cos de la resposta de la petició HTTP. Aquest tipus de bloqueig és més pesat que els anteriors, ja que hem d'analitzar tot el cos de la resposta per a determinar si hi ha bloqueig o no.
    e) Habilitar el bloqueig de contingut per a adults: aquesta opció de la configuració serà una opció per a activar o desactivar l'opció de mostrar pàgines amb contingut per a adults. La implementació d'aquest bloqueig serà molt semblant a la del punt anterior, amb la diferència que per a determinar si hi ha bloqueig o no haurem de comprovar si el contingut de la resposta de la petició http conté material per a adults o no. Per a fer aquesta comprovació farem servir una API d'un proveïdor extern. Concretament farem servir l'API de Datumbox (http://www.datumbox.com/machine-learning-api/). Farem una petició a aquesta API passant com a paràmetre el contingut del cos de la resposta de la petició http, i ens retornarà si té contingut per a adults o no. Segons aquesta resposta bloquejarem la petició o no. Aquest tipus de bloqueig serà el més pesat de tots, ja que a més de necessitar tota la resposta http, hem de consultar un servei extern.

2. Registrar l'activitat de navegació dels usuaris: aquest registre es podrà visualitzar i contindrà la següent informació:

    a) La URL de la petició
    b) Si s'ha bloquejat la petició o no, i el motiu del bloqueig en cas afirmatiu.
    c) La data i hora de la petició.

Per altra banda, l'aplicació tindrà els següents requisits no funcionals:

1. S'haurà de poder modificar la configuració dels paràmetres del filtratge i consultar el registre de navegació des de qualsevol lloc.

2. El procés de filtratge ha de ser transparent per a l'usuari (excepte en el cas obvi de que es produeixi un bloqueig).

3. El procés de filtratge no ha d'empitjorar l'experiència de navegació dels usuaris. Pot afectar a la velocitat de càrrega de les pàgines però sense que comporti una molèstia a l'usuari.

4. Anàlisi i disseny

En aquest capítol es descriurà el procés d'anàlisi i disseny del projecte.

4.1. Estudi dels diferents tipus d'aplicacions de filtratge web i selecció del tipus d'aplicació a desenvolupar

Entre totes les possibilitats i alternatives per a realitzar el projecte he contemplat la viabilitat d'aquests tipus d'aplicació:

- Aplicació instal·lable al dispositiu: consisteix en desenvolupar una aplicació específica que es pugui executar en un sistema operatiu concret, i que s'encarregui d'interceptar les peticions HTTP que es fan al sistema per a aplicar el filtratge.

- Servidor proxy: es un aplicació que s'executa en un servidor extern en comptes de al dispositiu de l'usuari. Aquest servidor centralitza el tràfic web dels dispositius que volem filtrar i analitza les peticions per realitzar el filtratge. Per a dirigir el tràfic dels dispositius que volem filtrar cap al servidor proxy hem de configurar els dispositius per a que apuntin a aquest servidor, o configurar la xarxa local per a que totes les peticions passin prèviament pel proxy abans de sortir a Internet.

- Extensió de navegador: és un tipus d'aplicació que funciona sobre un navegador web, i utilitza les APIs que el navegador posa a la seva disposició. Per a la nostra aplicació l'API principal que necessitaríem seria la API webRequest, que intercepta les peticions HTTP corresponents a la navegació que fa l'usuari amb el navegador web.

Tenint en compte els objectius marcats a l'inici del projecte m'he decantat per a implementar una extensió de navegador, pels següents motius:

- Facilitat d'instal·lació: una extensió de navegador és fàcil d'instal·lar i configurar en qualsevol dispositiu i sistema operatiu, ja que els navegadors més utilitzats (Google Chrome i Firefox) tenen versió per a tot tipus de sistemes operatius, tant d'escriptori com mòbils. A més, aquests navegadors son capaços de guardar la configuració dels usuaris i compartir-la entre tots els navegadors del mateix tipus que tinguin el mateix compte d'usuari, amb la qual cosa instal·lant l'extensió en un navegador es podrà fer servir en tots els navegadors que tinguin la mateixa sessió d'usuari. Per contra, si fem una aplicació instal·lable hauríem de fer una aplicació per a cada sistema operatiu on la volem fer servir. El cas del servidor proxy tampoc tindria aquesta facilitat, depenent de com ho configuréssim.

- Facilitat de configuració i accés al registre de navegació: en aquest cas totes tres opcions serien semblants, ja que si guardem la informació de configuració i el registre a l'aplicació hem d'accedir a l'aplicació, extensió

o proxy per a poder consultar o modificar aquesta informació. De tota manera una extensió de navegador tindria un avantatge, perquè com hem dit abans es comparteix la configuració entre els navegadors amb el mateix usuari, per la qual cosa podríem accedir des de qualsevol navegador que tingui la nostra sessió.

Relacionat amb l'últim punt, encara que una extensió de navegador permet consultar i modificar la informació, he decidit centralitzar en un servidor extern la informació d'usuaris i la seva configuració de filtratge i registre de navegació. L'accés i modificació d'aquesta informació es farà a través d'una API REST. D'aquesta manera si en un futur volem desenvolupar un portal web o aplicació mòbil per a gestionar aquesta informació, ho podrem fer a través de l'API REST i desenvolupar únicament la part frontal de l'aplicació en qüestió.

4.2. Rols d'usuari

A la nostra aplicació tindrem dos tipus diferents d'usuari:

- Usuari administrador: serà l'usuari que instal·larà i gestionarà l'aplicació. Podrà establir i modificar els paràmetres del filtratge web i consultar el registre de navegació. Per a poder fer aquesta gestió haurà de registrar-se a la nostra plataforma i establir una contrasenya.

- Usuari normal: seran els usuaris als que volem aplicar el filtratge web. No és necessari que aquests usuaris es registrin a la plataforma.

4.3. Model de dades de l’aplicació

El model de dades de l’aplicació contindrà dues entitats:

- Entitat Usuari: és l’entitat que modelarà els usuaris administradors de l’aplicació. Contindrà el mail i password per identificar els usuaris, i la informació de la configuració.

- Entitat Event: és l’entitat que contindrà la informació de totes les visualitzacions fetes pels usuaris.

El diagrama del model de dades és el següent:

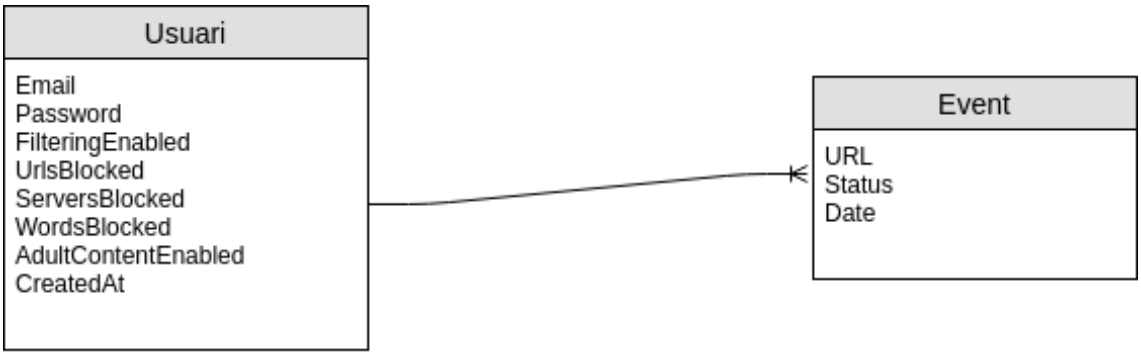


Figura 7: Model de dades

4.4. Diagrames de casos d’ús

A l’aplicació tenim clarament diferenciats dos actors diferents:

- Usuari administrador: és qui instal·la i administra la configuració de l’aplicació. També pot consultar la navegació que han realitzat els usuaris de l’aplicació. A més, l’usuari administrador si utilitza el navegador per navegar per internet és com si fos un usuari normal.

- Usuari del navegador: és l’usuari que utilitza el navegador, a qui se li aplica la la configuració de l’extensió durant la seva navegació.

Els casos d’ús de l’aplicació seran els següents:

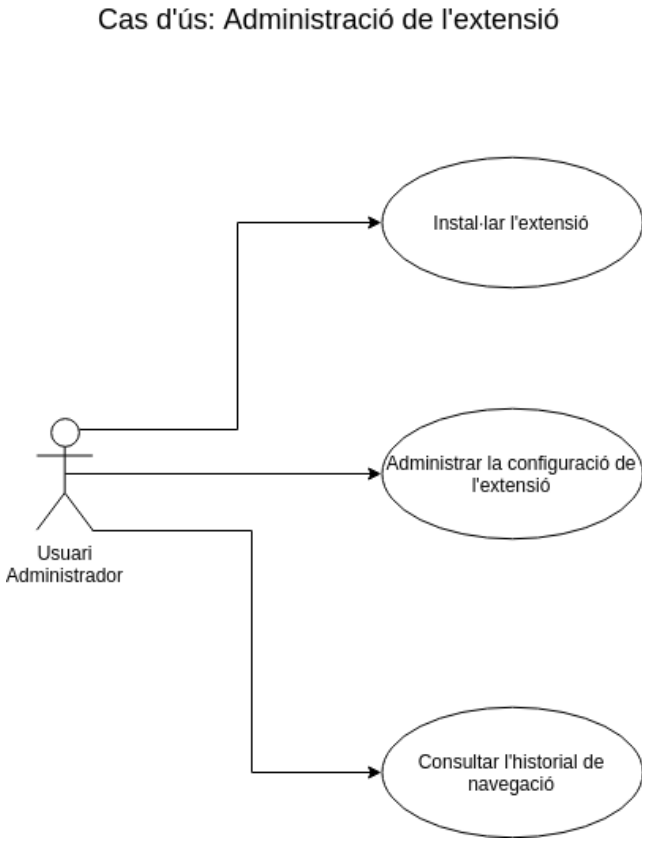


Figura 8: Cas d'ús administració de l'extensió

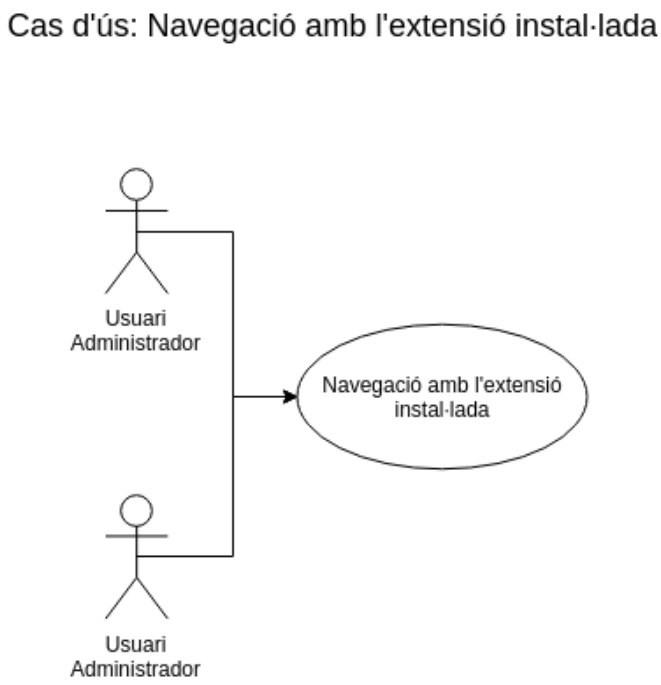


Figura 9: Cas d'ús navegació amb l'extensió instal·lada

4.5. Diagrama de flux de l’aplicació

El diagrama de flux de l’aplicació serà el següent:

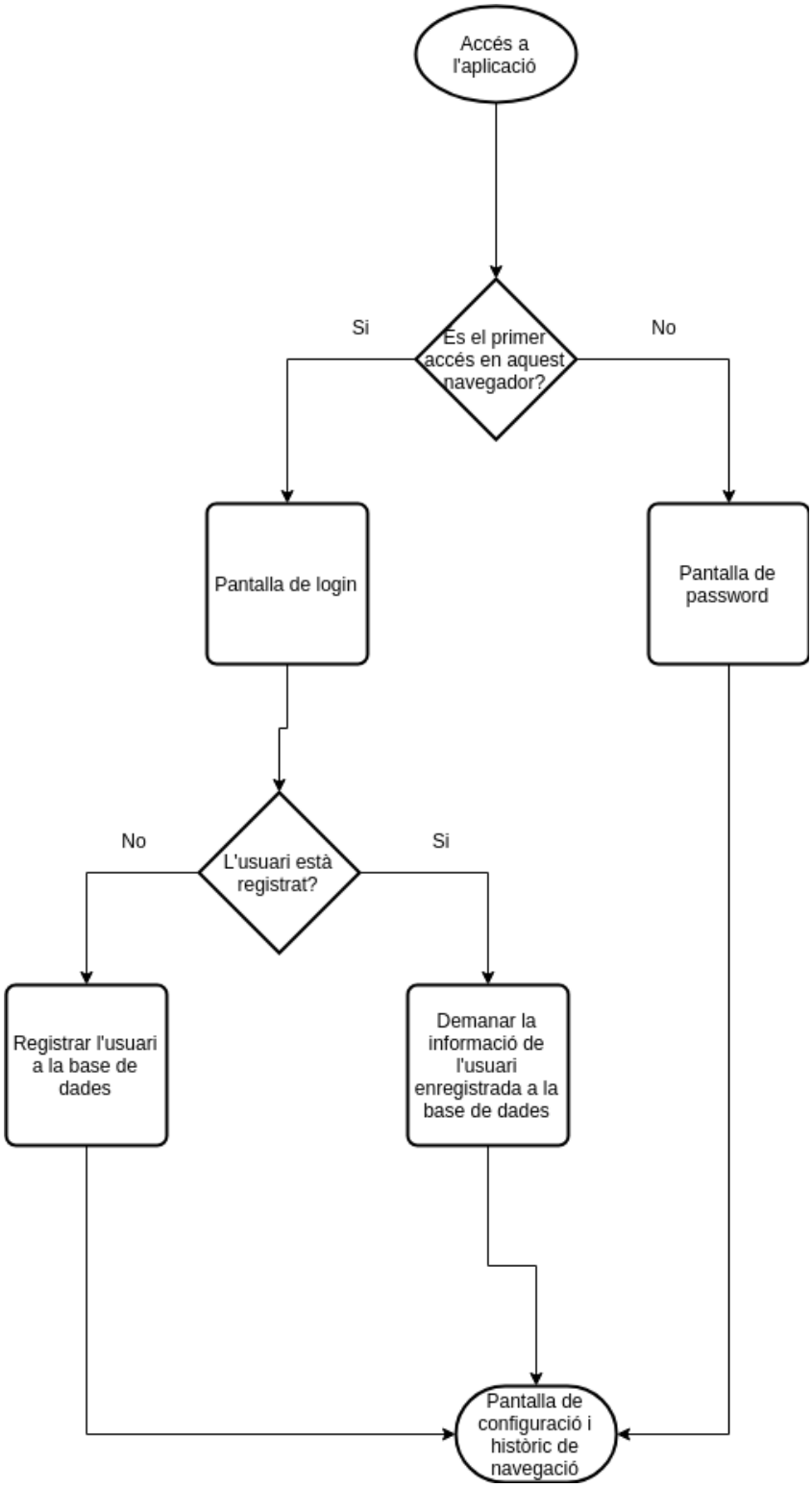


Figura 10: Diagrama de flux de l'aplicació

4.6. Wireframes de l’aplicació

Els dissenys inicials de les pantalles de la part d’administració de l’aplicació (wireframes) son els següents:

- Pantalla de login:

Figura 11: Wireframe de la pantalla de login

- Pantalla d’inserció de password:

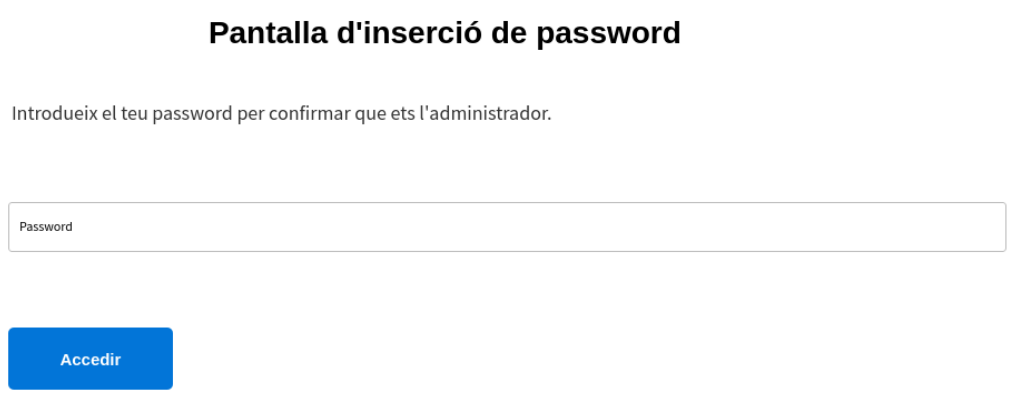


Figura 12: Wireframe de la pantalla d'inserció de password

- Pantalla de configuració i històric de navegació:

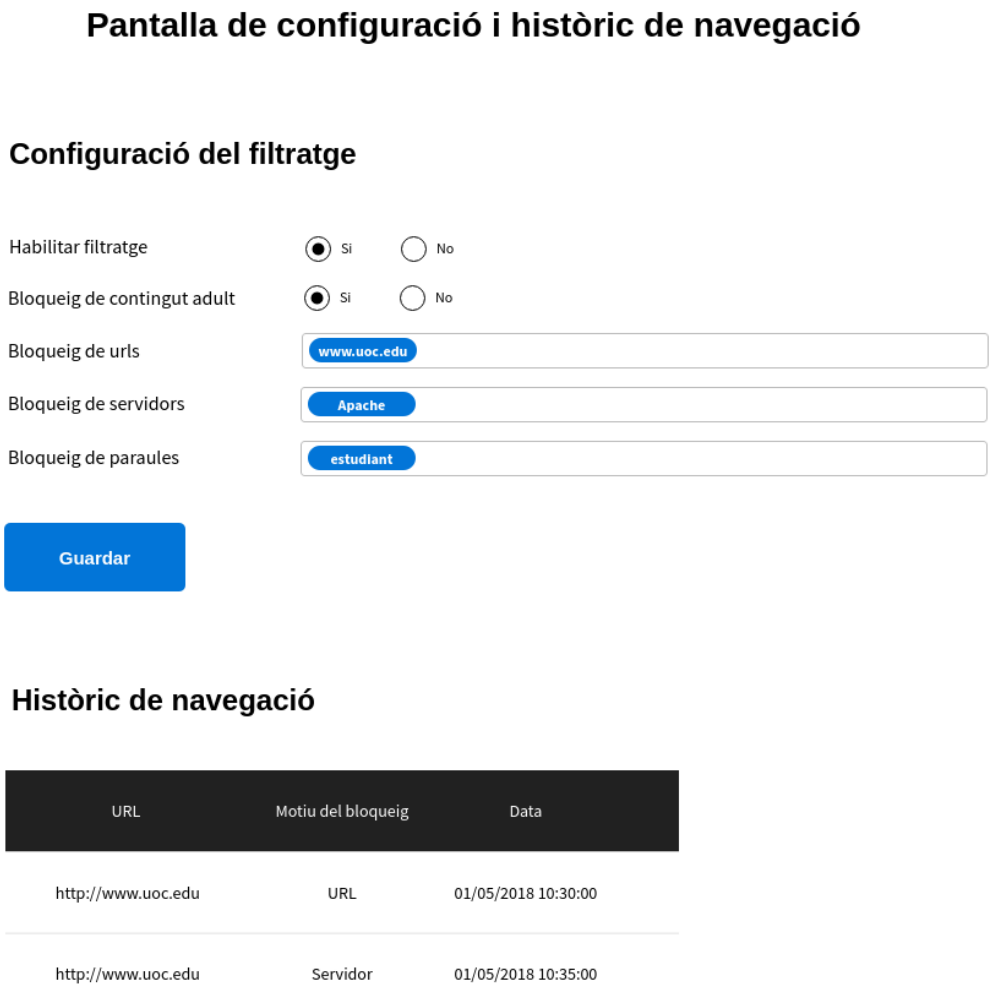


Figura 13: Wireframe de la pantalla de configuració i registre de navegació:

Per altra banda, també necessitem el disseny de la pantalla que es mostrarà als usuaris del navegador quan es bloquegi l'accés a una pàgina web.

- Pantalla de bloqueig d'accés a una pàgina web:

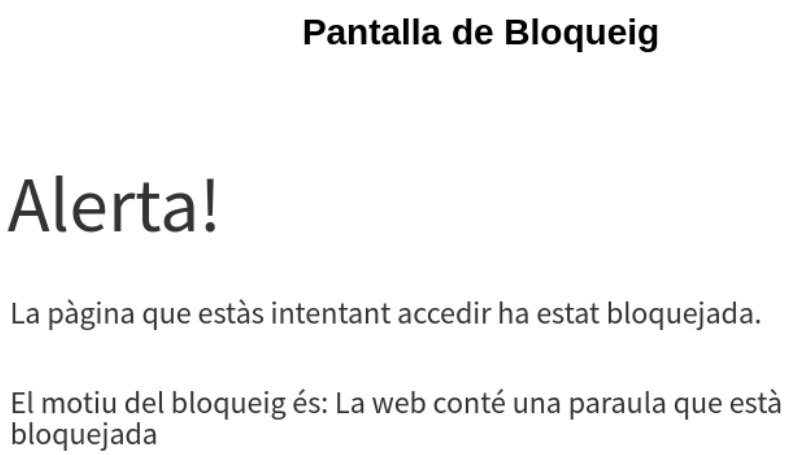


Figura 14: Wireframe de la pantalla de bloqueig

5. Implementació

En aquest apartat parlarem del procés d'implementació de l'aplicació de filtratge web.

5.1. API per a interceptar les peticions HTTP

Per al desenvolupament de l'extensió del navegador farem servir l'API webRequest del navegador. Aquesta API defineix una sèrie d'esdeveniments que segueixen el cicle de vida d'una petició web, que són els següents:

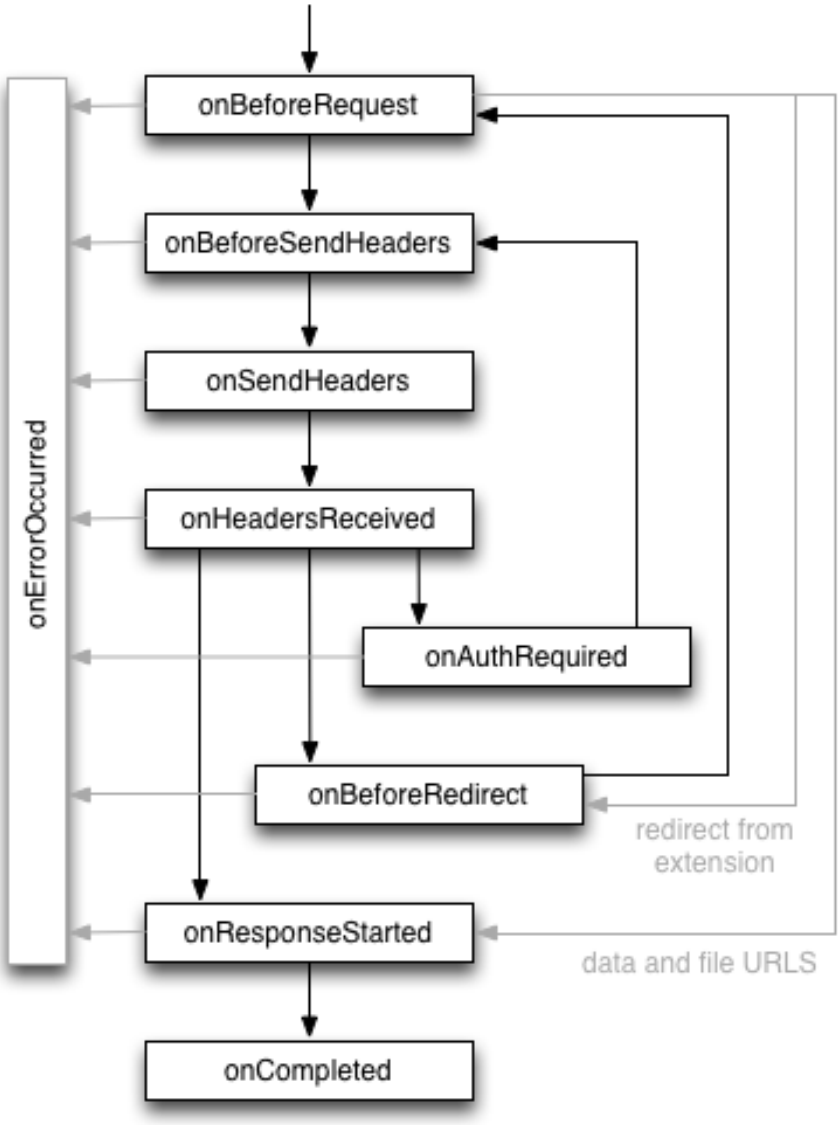


Figura 15: Esdeveniments de la API de webRequest

Els esdeveniments que farem servir per a assegurar-nos que es compleixen les regles que ha definit l'usuari administrador a la configuració són:

- onBeforeRequest: esdeveniment que farem servir per a verificar el filtratge per domini. En aquest esdeveniment podem obtenir la URL de la petició i comparar-la amb el llistat de dominis als quals no es vol permetre l'accés. En aquest esdeveniment també inicialitzarem la funció que farà el filtratge de les respostes HTTP per poder analitzar el cos de les respostes i poder detectar si conté alguna paraula que estigui al filtre de paraules a bloquejar o si té contingut per a adults.

- onHeadersReceived: esdeveniment que farem servir per a verificar el filtratge per servidor web. En aquest esdeveniment podem revisar els headers inclosos en la resposta del servidor, obtenir el header que conté la informació del tipus de servidor i comparar-lo amb el llistat de servidors que es volen bloquejar.

La idea inicial era desenvolupar una extensió per a Google Chrome en comptes de per al navegador Firefox, ja que aquest és el navegador que utilitzo normalment. Però quan vaig començar el desenvolupament vaig trobar que la implementació que feia de l'API webRequest no tenia inclosa la funció filterResponseData (https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest/filterResponseData) que és on es permet analitzar el cos de la resposta http per a implementar els filtres sobre el contingut de la resposta.

Vaig buscar alternatives a la utilització d'aquesta funció, una d'elles era utilitzar l'API chrome.debugger que permet interceptar la resposta del servidor, però era més complicada d'utilitzar i a més mostra una barra groga al navegador que empitjora l'experiència d'usuari.

Finalment vaig decidir utilitzar el navegador Firefox, ja que implementa la funció que necessitava i no afecta en res en l'objectiu final de la utilització de l'aplicació de filtratge web.

5.2. Arquitectura de l’aplicació

L’aplicació de filtratge web constarà dels següents elements:

- Una base de dades per emmagatzemar les dades dels usuaris, la configuració d’aquests usuaris, i la informació de les pàgines visitades. La base de dades serà una base de dades MongoDB. El model de dades que farem servir per guardar la informació s’especifica al punt 3.6.

- Servidor amb una API REST que oferirà diferents endpoints per a crear/loguejar usuaris, actualitzar la configuració dels usuaris, guardar i llistar els esdeveniments de navegació, etc. La API REST es construirà amb NodeJS juntament amb el framework Express, i es farà servir Nginx per a configurar un proxy invers que enllaci el port del servidor del procés de NodeJS amb el port 80 del Nginx.

- Extensió del navegador Firefox amb una pàgina d’administrador que contindrà el dashboard per a permetre l’edició de la configuració i visualitzar el llistat de pàgines visitades.

El diagrama de l’arquitectura de l’aplicació és el següent:

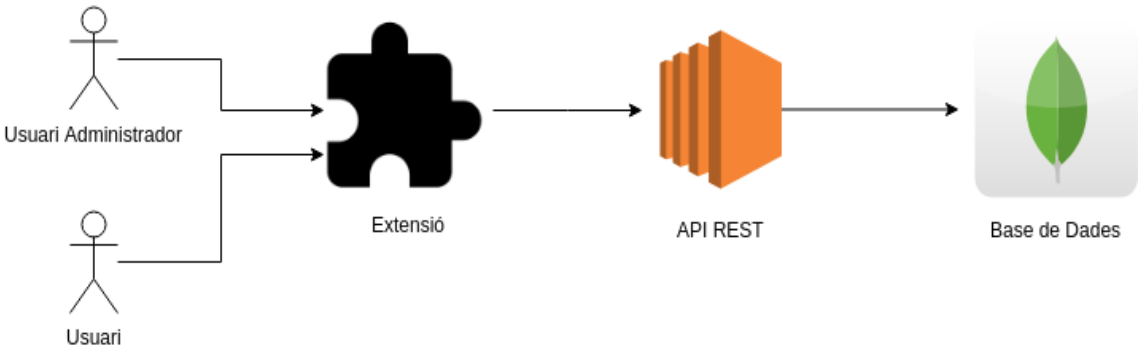


Figura 16: Diagrama d'arquitectura

5.3. Base de dades

La base de dades escollida per a emmagatzemar les dades de l'aplicació ha estat MongoDB. Com que MongoDB és una base de dades schemaless (és a dir, no necessita que definim una estructura fixa), no és necessari que executem cap instrucció a MongoDB per a definir l'estructura de les col·leccions ni dels documents de la base de dades. De tota manera, si que tenim un model de dades que hem definit durant el disseny de l'aplicació, però el fixarem a la implementació de l'API REST que veurem a continuació.

5.4. API REST

Per a implementar l'API REST hem fet servir NodeJS juntament amb el framework ExpressJS. ExpressJS ens permet crear d'una manera ràpida i relativament senzilla una API REST amb les funcionalitats bàsiques que ens serviran per al propòsit de la nostra aplicació.

Per a la connexió i comunicació amb la base de dades hem utilitzat el mòdul Mongoose. Aquest mòdul ens permet definir un esquema per als documents de la base de dades. Aquest és el mòdul que hem fet servir per a definir el model de dades, com s'ha comentat a l'apartat anterior. Els models que hem definit mitjançant Mongoose son els següents:

- User: un model per a definir els usuaris, on guardem l'e-mail, la contrasenya (encriptada), la configuració del filtratge, i la data de creació de l'usuari. En aquest model també hem afegit les funcions per a encriptar la contrasenya quan es crea un usuari nou, i per a comparar la contrasenya rebuda amb la contrasenya de l'usuari i validar si és correcta. Per a la lògica relativa a l'encriptació hem fet servir el mòdul bcrypt.

- Event: un model per a definir com guardem els esdeveniments de la navegació del usuaris. Els esdeveniments fan referència a les pàgines visitades pels usuaris, i guardem la pàgina a la que s'ha accedit o intentat accedir, l'usuari que administra l'extensió del navegador que s'ha utilitzat, si s'ha bloquejat o no (i el motiu del bloqueig), i la data i hora de l'accés.

La implementació de la API REST la hem fet al fitxer server.js. En aquest fitxer s'ha fet el següent:

- Fer servir la api de ExpressJS per a implementar la funcionalitat que gestionarà les peticions a la nostra API REST.

- Aixecar el servidor i fer la connexió a la base de dades.

Els endpoints que oferirà seran els següents:

1. POST /user
   a) Crea un usuari administrador en cas que l'e-mail no existeixi a la base de dades, o fa el login d'usuari en cas que existeixi l'e-mail.
   b) Body: { e-mail, password }
   c) Resposta:
      i. Status = 201 en cas que l'usuari no existeixi
      ii. Status = 200 en cas que l'usuari existeixi i el password sigui correcte
      iii. Status = 401 en cas que l'usuari existeixi i el password sigui incorrecte
      iv. Status = 500 si s'ha produït un error

2. PATCH /user/{userId}
   a) Actualitza la configuració de l'usuari amb id = userId.
   b) Body: { config }
   c) Resposta:
      i. Status = 204 en cas que s'hagi actualitzat correctament
      ii. Status = 500 si s'ha produït un error

3. GET /user/{userId}
   a) Retorna la informació de la configuració de l'usuari amb id = userId.
   b) Resposta:
      i. Status = 200 en cas que es retorni la informació correctament
      ii. Status = 500 si s'ha produït un error

4. POST /event/{userId}
   a) Registra un esdeveniment de visualització a la base de dades.
   b) Body: { URL, userId, status }
   c) Resposta:
      i. Status = 201 si s'ha registrat l'esdeveniment correctament
      ii. Status = 500 si s'ha produït un error

5. GET /events/{userId}
   a) Retorna tots els esdeveniments de l'usuari amb id = userId
   b) Resposta:
      i. Status = 200 en cas que es retornin els esdeveniments correctament
      ii. Status = 500 si s'ha produït un error

5.5. Extensió del navegador

L'extensió del navegador la hem desenvolupat seguint les directrius especificades a la documentació de Mozilla sobre el desenvolupament d'extensions per al navegador Firefox.

En primer lloc hem creat el fitxer manifest.json que conté les metadades que necessita l'extensió per poder funcionar correctament. En el cas de la nostra extensió, aquest fitxer conté el següent:

- El nom
- La versió
- La descripció
- Els permisos que necessita, que en el nostre cas són:
  - webRequest: per a poder accedir a l'api webRequest amb la qual podem accedir a les peticions http que es realitzen al navegador.
  - webRequestBlocking: per a poder bloquejar les peticions http.
  - <all_urls>: necessari per a indicar que volem inspeccionar totes les peticions http.
  - storage: per poder guardar informació al local storage del navegador.
- Els scripts que executa en segon pla, que en el nostre cas són:
  - background.js: el fitxer que conté tota la lògica de filtratge web.
  - utils.js: un fitxer on hem extret funcionalitat comuna.
- La configuració de la pàgina d'opcions (la nostra pàgina d'administració)
- Les pàgines HTML a les que pot accedir, que en el nostre cas és la pàgina amb el missatge de bloqueig.
- El nom dels fitxer d'imatge que contenen la icona de l'extensió.

El fitxer background.js és el que conté tota la lògica per a fer el filtratge web:

- En primer lloc comprova si s'acaba d'instal·lar l'extensió i, en cas afirmatiu, obre la pàgina d'administració. D'aquesta manera l'usuari administrador por establir la configuració inicial del filtratge.
- També hi ha un escoltador de l'esdeveniment clic del botó de l'extensió, per a mostrar la pàgina d'administració si un usuari fa clic al botó.
- La resta del codi d'aquest fitxer és l'encarregat de llegir la configuració de filtratge a través de la nostra API REST i de fer servir la api webRequest del navegador per interceptar les peticions http. Concretament es fan servir els següents mètodes de webRequest:
  - onBeforeRequest: aquest mètode és el que llegeix la petició request abans de ser enviada al servidor. És aquí on fem la petició a la nostra

API REST per a llegir la configuració de filtratge. Consultem la configuració de filtratge cada vegada que es fa una petició per assegurar-nos que està actualitzada, ja que l'administrador pot canviar la configuració en qualsevol moment. Comprovem si la URL de la petició coincideix amb alguna de les urls a bloquejar que hi han configurades, i en cas afirmatiu es bloqueja la petició.
En aquest mètode també s'afegeix un filtre que revisarà la resposta del servidor quan aquesta arribi. Dintre d'aquest filtre es revisarà per una banda el contingut del cos de la resposta per comprovar si conté alguna paraula de les que hi han configurades al filtre per a bloquejar, i per altra banda es comprovarà si conté contingut destinat per a adults. Per a fer la detecció de contingut adult utilitzem el servei extern oferit per datum (http://www.datumbox.com/machine-learning-api/), fent una petició a la API REST que ofereixen.

- onHeadersReceived: aquest mètode s'executa quan es reben les capçaleres de la resposta de la petició http. Llavors revisem si hi ha alguna capçalera amb la clau server, i en cas afirmatiu es revisa si el valor d'aquesta capçalera coincideix amb algun dels servidors a bloquejar de la configuració de filtratge. Si hi ha coincidència es bloqueja la petició.

La gestió de la pàgina d'administració es fa a través del fitxers options.html i options.js:

- El fitxer options.html conté el HTML de la pàgina d'administració. Per a fer la maquetació hem fet servir la llibreria bootstrap. En aquesta pàgina hem creat tres capes diferents que s'oculten o es mostren segons el que es vol mostrar a l'usuari. La primera capa conté el formulari per a fer el login/registre dels usuaris. La segona capa conté el formulari per a introduir la contrasenya quan l'usuari ja ha fet login i vol entrar a la pàgina d'administració. Per últim, la tercera capa conté el HTML que correspon a la pàgina d'administració. Aquesta capa està dividida en dues parts, la primera conté la gestió de la configuració del filtratge web, i la segona conté l'historial de navegació dels usuaris.

- El fitxer options.js és el que gestiona la lògica de la pàgina d'administració. Per una banda gestiona quan s'ha de mostrar cadascuna de les tres capes, segons si ja hi ha un usuari guardat al storage del navegador. Per una altra banda s'encarrega de permetre la consulta i modificació de la configuració del filtratge web, fent les corresponents crides a la API REST. Per últim també s'encarrega de consultar l'historial de navegació i mostrar el resultat a la corresponent taula, a més de fer la recarrega si l'usuari ho demana.

L’últim component important de la implementació de l’extensió és la pàgina que es mostra quan hi ha un bloqueig. Els fitxers que corresponen a aquesta pàgina són els següents:

- blockpage.html: conté el HTML de la pàgina de bloqueig. Com a la pàgina d’administració, hem fet servir la llibreria bootstrap per a fer la maquetació.

- blockpage.js: únicament s’encarrega d’obtenir el paràmetre que indica el motiu del bloqueig per a mostrar el missatge corresponent a l’usuari.

6. Instal·lació i configuració de l'entorn de producció

Per a poder fer accessible de forma real la nostra aplicació he procedit a configurar un entorn de producció que permeti la seva utilització per qualsevol usuari.

6.1. Instal·lació i configuració del servidor

Per a aixecar un servidor i instal·lar tot el programari necessari que contingui la base de dades i l'API REST he seguit els següents passos:

1. Obrir un compte d'AWS a través de la següent URL: https://aws.amazon.com/es/

2. Aixecar una instància d'EC2 amb el sistema operatiu Ubuntu 16.04. En aquest procés haurem de fer:

    a) Configurar el security group de la instància per a permetre els següents accessos:

        i. Accés SSH des de la nostra IP. Ho necessitarem per a accedir mitjançant SSH per a instal·lar tot el programari que necessitem al servidor i configurar la màquina.
        ii. Accés al port 80 des de qualsevol IP. Ho necessitarem perquè la nostra API REST que servirà el servidor sigui accessible per a qualsevol usuari.

    b) Demanar una 'Elastic IP' i associar-la a la nostra instància. D'aquesta manera tindrem una IP pública fixa que ens servirà per a apuntar a la API REST.

3. Una vegada estigui disponible la instància, ens connectarem mitjançant SSH per a instal·lar el següent programari:

    a) NodeJS: la API està construïda amb el llenguatge NodeJS i per tant necessitarem instal·lar-lo. Per fer-ho executarem les següents instruccions:

        i. curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
        ii. sudo apt-get update
        iii. sudo apt-get install nodejs
        iv. node -v (per comprovar que s'ha instal·lat correctament)

    b) PM2: és un gestor de processos per a NodeJS, i el farem servir per a gestionar el procés del servidor encarregat de l'API. Haurem de fer el següent:

i. sudo npm install pm2 -g
ii. pm2 -v (per comprovar que tot ha anat bé)

c) Nginx: necessitem instal·lar i configurar Nginx per tal de configurar un proxy invers que enllaci el port que utilitza el procés de NodeJS amb el port 80. No podem utilitzar directament el port 80 ja que Linux restringeix l'ús dels ports inferiors a 1024 als processos que no tenen permisos root. Per a fer la instal·lació del Nginx i la seva configuració com a proxy invers seguirem el següent procediment:

i. sudo apt-get install nginx
ii. systemctl status nginx (per comprovar que s'ha instal·lat correctament i el servei s'ha iniciat)
iii. Esborrem el fitxer de configuració per defecte de Nginx amb sudo rm /etc/nginx/sites-available/default
iv. Crearem un nou fitxer de configuració de Nginx amb la comana sudo vim /etc/nginx/sites-available/api i el deixarem de la següent manera:

```
server {
    listen 80;
    server_name example.com;

    location / {
        proxy_set_header   X-Forwarded-For $remote_addr;
        proxy_set_header   Host $http_host;
        proxy_pass         "http://127.0.0.1:1337";
    }
}
```

v. Creem l'enllaç a sites-enabled amb sudo ln -s /etc/nginx/sites-available/api /etc/nginx/sites-enabled/api
vi. Reiniciem Nginx amb sudo systemctl restart nginx

d) MongoDB: és la base de dades que utilitzem per guardar la informació dels usuaris i els esdeveniments. Per instal·lar-la haurem d'executar les següents comandes:

i. sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
ii. echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
iii. sudo apt-get update
iv. sudo apt-get install mongodb-org
v. Hem de crear un fitxer per administrar el servei de MongoDB. Per això fem sudo vim /etc/systemd/system/mongodb.service, i escrivim el següent al fitxer:

```
[Unit]
Description=High-performance, schema-free document-oriented database
After=network.target

[Service]
User=mongodb
ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf

[Install]
WantedBy=multi-user.target
```

vi. sudo systemctl start mongodb
vii. sudo systemctl enable mongodb
viii. sudo systemctl status mongodb (per comprovar que el servei s'ha iniciat correctament)

e) Configuració del projecte:

i. En primer lloc anem al directori /var/www i clonem el projecte: sudo git clone https://josejuansanz@bitbucket.org/uoctfg/server.git
ii. Canviem el propietari del directori del projecte amb sudo chown -R ubuntu: server
iii. Instal·lem totes les dependències amb npm install
iv. Iniciem el servidor amb pm2 mitjançant la comanda pm2 start server.js --name api

Una vegada finalitzat tot aquest procès, ja tindrem la nostra API REST funcionant en un servidor accessible des de qualsevol màquina.

6.2. Empaquetar i publicar l'extensió de Firefox

Perquè l'extensió estigui disponible per a tots els usuaris hem d'empaquetar-la i publicar-la. Per fer-ho seguirem els següents passos:

1. Primer de tot hem de posar la IP del servidor que hem aixecat al codi de l'extensió que fa les peticions a l'API REST per assegurar-nos que les peticions s'envien correctament.

2. Empaquetar els fitxers de l'extensió, des del directori on tenim el codi hem d'executar la comanda zip -r -FS ../web-filtering-app.zip *

3. Hem de crear un compte de desenvolupador a la URL https://developer.mozilla.org/en-US/Add-ons/Distribution

4. Hem de verificar el compte a través d'un enllaç que ens arribarà a l'e-mail amb el que ens hem registrat.

5. A continuació hem d'anar al link https://addons.mozilla.org/en-US/developers/ i clicar a Submit Your First Add-on.

6. El següent pas dóna l'opció d'escollir entre publicar l'extensió al lloc web de Firefox (on this site) o permetre la instal·lació a través d'un enllaç gestionat per tu. Escollim la segona opció.

7. A continuació ens apareix la pantalla que ens permet pujar el zip que hem generat al primer pas.

8. Quan finalitza la pujada del fitxer ens apareix el botó per a signar l'extensió.

9. Una vegada signat s'acaba el procés i ens permet baixar el fitxer signat amb el qual podrem instal·lar l'extensió al navegador que desitgem.

10. Per fer la instal·lació, obrim Firefox i escrivim about:addons a la barra de navegació. En aquesta pantalla fem clic al select de configuració que hi ha a la part superior dreta i seleccionem Install add-on from file. Seleccionem el fitxer que hem descarregat a l'apartat anterior i l'extensió s'instal·larà.

11. Per poder fer accessible l'extensió a tothom a qui li vulguem passar, pujarem el fitxer que acabem de descarregar al servei d'emmagatzemament anomenat Simple Storage Service (S3) d'Amazon Web Services. D'aquesta manera qualsevol persona a que li passem el link el podrà descarregar, i podrem tenir un control de qui l'està utilitzant.

7. Manual d’ús de l’aplicació

L’únic requisit que necessitem per fer servir l’aplicació de filtratge web és tenir instal·lat el navegador web Firefox.

El primer que haurem de fer és instal·lar l’extensió al navegador Firefox a on vulguem aplicar el filtratge en la navegació dels usuaris.

Podem baixar l’extensió des del següent link https://s3.amazonaws.com/web-filtering-app/aplicacio_de_filtratge_de_pagines_web-1.0.3-an%2Bfx.xpi, que és la URL del fitxer que hem pujat al servei S3 prèviament. Una vegada tenim el fitxer descarregat al nostre ordinador obrim el navegador Firefox i escrivim a la barra de navegació about:addons. Es carregarà la pantalla d’administració d’extensions del navegador, des d’on podrem fer clic al selector de configuració que hi ha a la part superior dreta i seleccionar l’opció Install add-on from file. A la finestra que s’obrirà hem de seleccionar el fitxer que hem descarregat prèviament, i el navegador instal·larà l’extensió.

El primer que fa l’extensió un cop instal·lada és obrir la finestra d’opcions. En aquesta pantalla el primer que se’ns demanarà serà introduir el nostre usuari i contrasenya. Si és la primera vegada que fem servir l’aplicació haurem d’introduir el nostre e-mail i la contrasenya que volem fer servir, i ens crearà un usuari nou. Si ja tenim un usuari creat prèviament, haurem d’introduir el nostre e-mail i contrasenya. En cas que la contrasenya introduïda no sigui correcta l’aplicació no ens deixarà entrar i ens mostrarà un missatge d’error.

Una vegada hem fet el login correctament es mostrarà la pantalla d’administració de l’aplicació. Aquesta pantalla està dividida en dues parts: la part d’administració de la configuració de filtratge i la part de visualització de l’historial de navegació.

A la part d’administració de la configuració de filtratge és on podem establir els paràmetres en els quals es basarà l’aplicació per a realitzar el filtratge de la navegació web.

Els paràmetres de configuració que hi ha disponibles són:

- Habilitar el filtratge web: aquesta opció habilitarà o deshabilitarà el filtratge de la navegació. Si el filtratge web està activat s’aplicarà el bloqueig a la navegació, i si està desactivat es permetrà navegar lliurement.

- Bloquejar el contingut adult: si aquesta opció està activada es bloquejaran les webs que es detecti que continguin contingut destinat a adults.

- Bloqueig d’urls: dintre d’aquest camp es poden incloure els dominis que volem que es bloquegin.

- Bloqueig de servidors web: a aquest camp es poden escriure els noms dels servidors que volem bloquejar.

- Bloqueig de paraules: aquí es poden indicar totes les paraules que volem filtrar. Si alguna de les paraules d'aquest camp apareixen al contingut de la pàgina web a la que s'ha intentat accedir, aquesta pàgina es bloquejarà.

A la part d'històric de navegació apareixerà una taula amb tota la informació de les pàgines que s'han accedit o intentat accedir.

La taula conté la següent informació:

- URL: la URL de la web a la que s'ha accedit.

- Motiu del bloqueig: el motiu pel qual s'ha bloquejat l'accés a la web.

- Data: dia i hora en la qual s'ha realitzat l'accés.

També es pot accedir a aquesta pantalla d'administració a través del botó de l'extensió, que haurà aparegut a la part superior dreta del navegador, a la zona on es col·loquen els botons d'extensió. Al fer clic en aquest botó s'obrirà una nova pestanya, demanant en primer lloc introduir el password de l'usuari administrador. D'aquesta manera, només l'usuari administrador podrà modificar la configuració i consultar l'històric de navegació.

Una vegada introduït la contrasenya de l'usuari administrador s'obrirà la pantalla d'administració.

Podem instal·lar l'extensió a tants navegadors Firefox com vulguem i, si fem login amb el mateix usuari, la configuració del filtratge es compartirà entre tots ells. Igualment, l'històric de navegació conjunta.

En quant a un usuari del navegador amb l'extensió instal·lada, la seva navegació estarà condicionada a la configuració que hagi establert l'usuari administrador. Si alguna de les webs que visita compleixen alguns dels criteris configurats al filtratge per l'usuari administrador, no es permetrà l'accés a la web i es mostrarà una pantalla amb un missatge indicant que s'ha bloquejat l'accés, i el motiu pel qual s'ha bloquejat.

8. Proves de l’aplicació

S’ha realitzat una bateria de proves per a comprovar el correcte funcionament de l’aplicació de filtratge web. Les proves s’han fet amb un navegador Firefox executat sobre el sistema operatiu Ubuntu 16.04.

Les proves realitzades han estat les següents:

- Instal·lar l’extensió a partir de l’enllaç de S3 on hem pujat l’instal·lable de l’extensió, tal i com s’explica a l’apartat anterior Manual d’us de l’aplicació. S’obrirà automàticament la pantalla d’administració, demanant el login d’usuari. També s’afegirà un nou botó a la barra d’eines que ens permetrà accedir a l’administració de l’aplicació fàcilment.

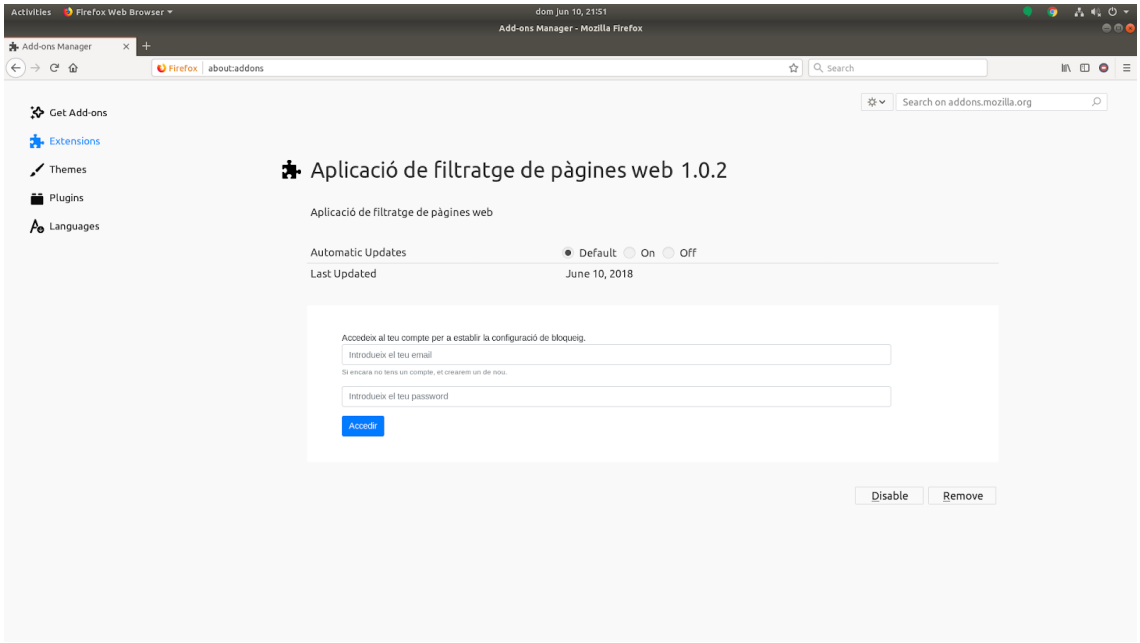


Figura 17: Prova d’instal·lació i càrrega de la pantalla de login

- Creem un compte nou amb un e-mail no registrat prèviament que ens portarà a la pantalla d'administració.

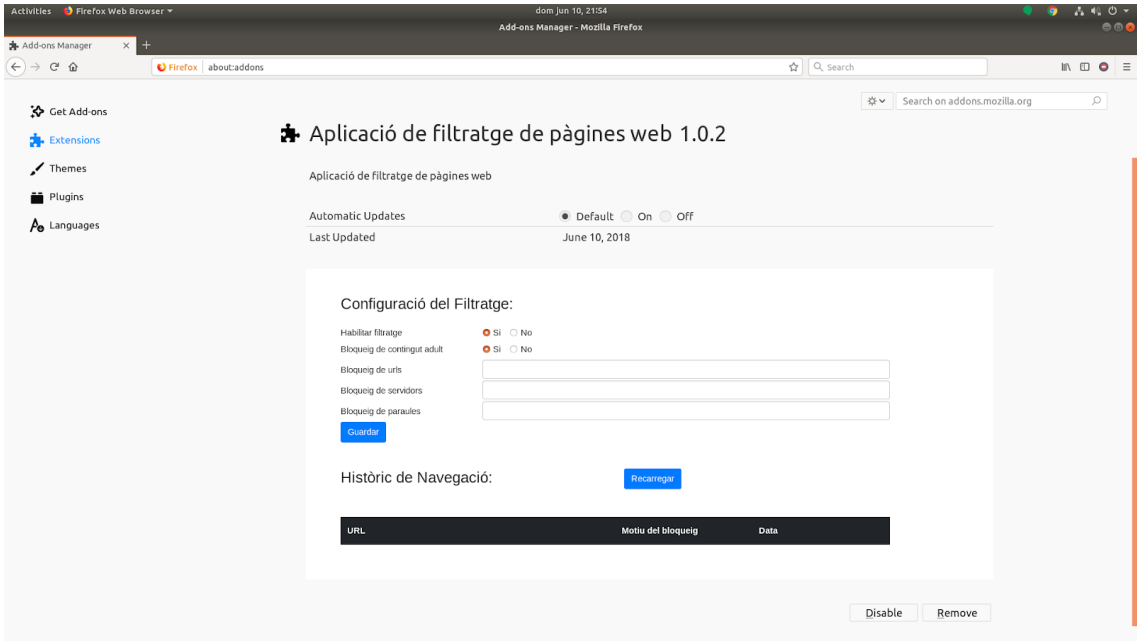


Figura 18: Prova de login/registre inicial

- Per provar que el bloqueig per URL funciona correctament, afegim al camp Bloqueig de urls la URL uoc.edu i fem clic a Guardar. Seguidament obrim una nova pestanya al Firefox i intentem accedir al web de la UOC.

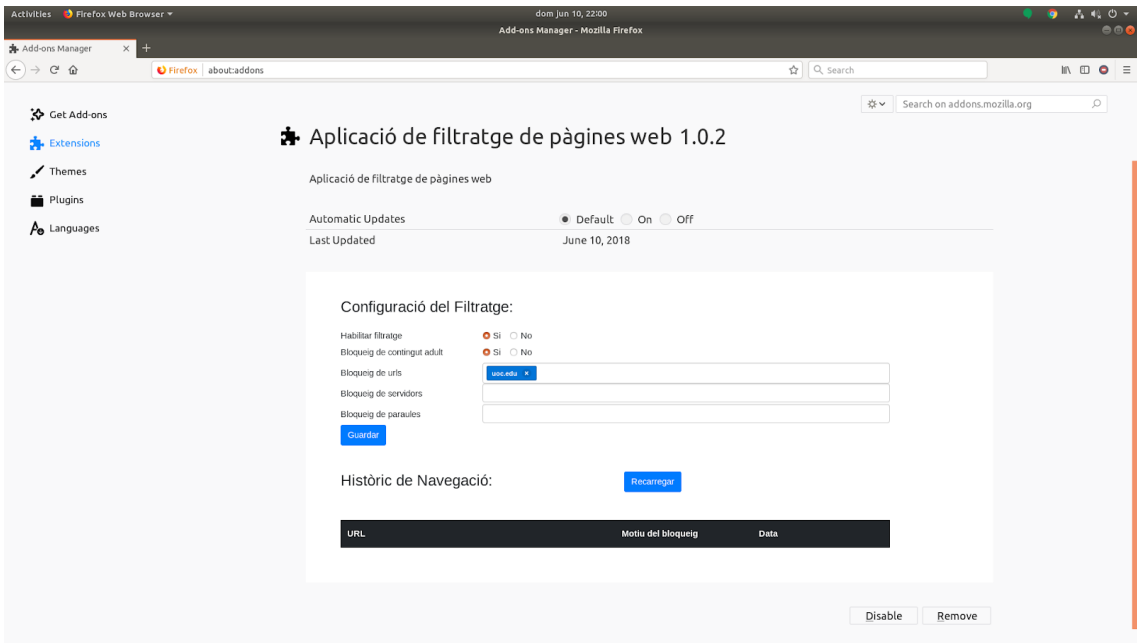


Figura 19: Prova de configuració del bloqueig per URL

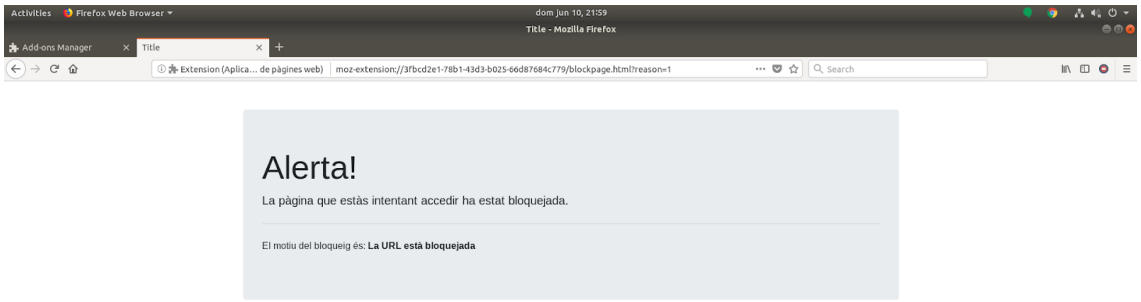


Figura 20: Prova del bloqueig per URL

- Una vegada comprovat que el filtre està funcionant, provem a desactivar el filtratge web canviant l'opció Habilitar filtratge a No, i provem a entrar de nou al web de la UOC.

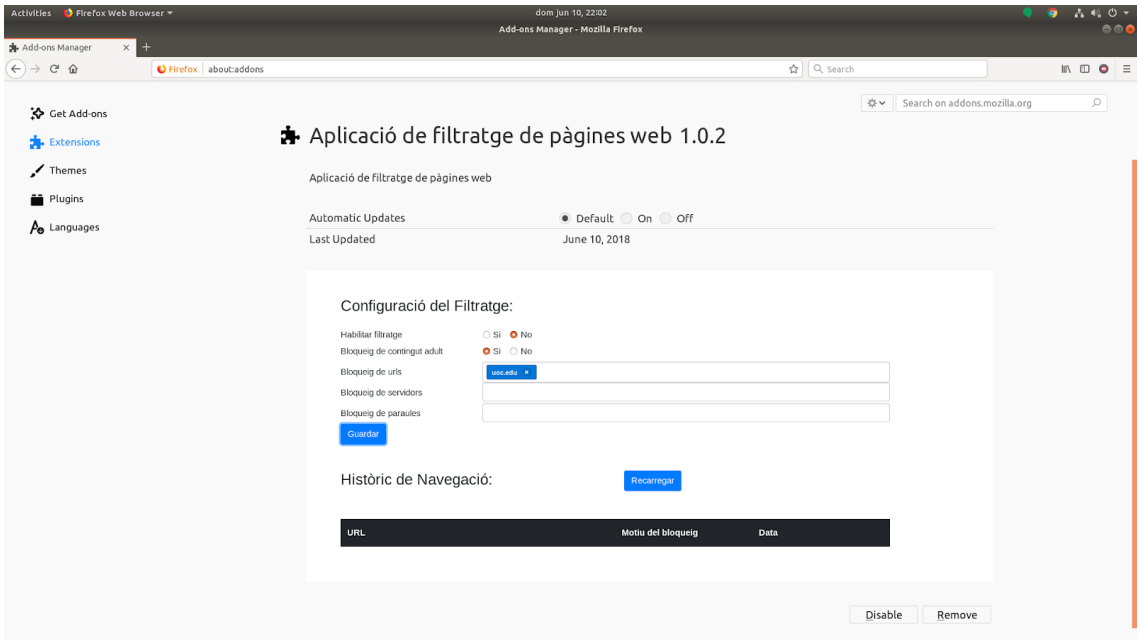


Figura 21: Prova de desactivació del filtratge web

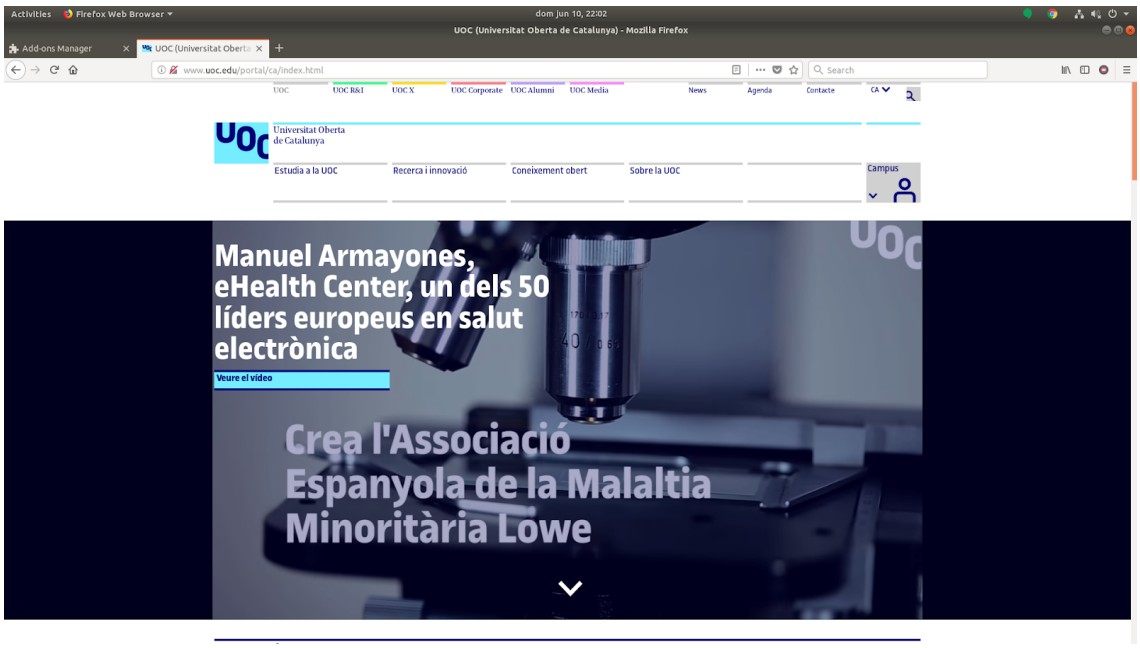


Figura 22: Prova d'accés amb el filtre desactivat

- Procedim a provar el filtratge per tipus de servidor. Amb l'ajut de la consola del Firefox revisem els headers que ens retorna el servidor de la UOC i esbrinem que el servidor que utilitza el web de la UOC és Apache. Ho afegim al camp Bloqueig de servidors de la configuració (tornant a habilitar el filtratge que havíem desactivat prèviament) i provem a accedir de nou al web.

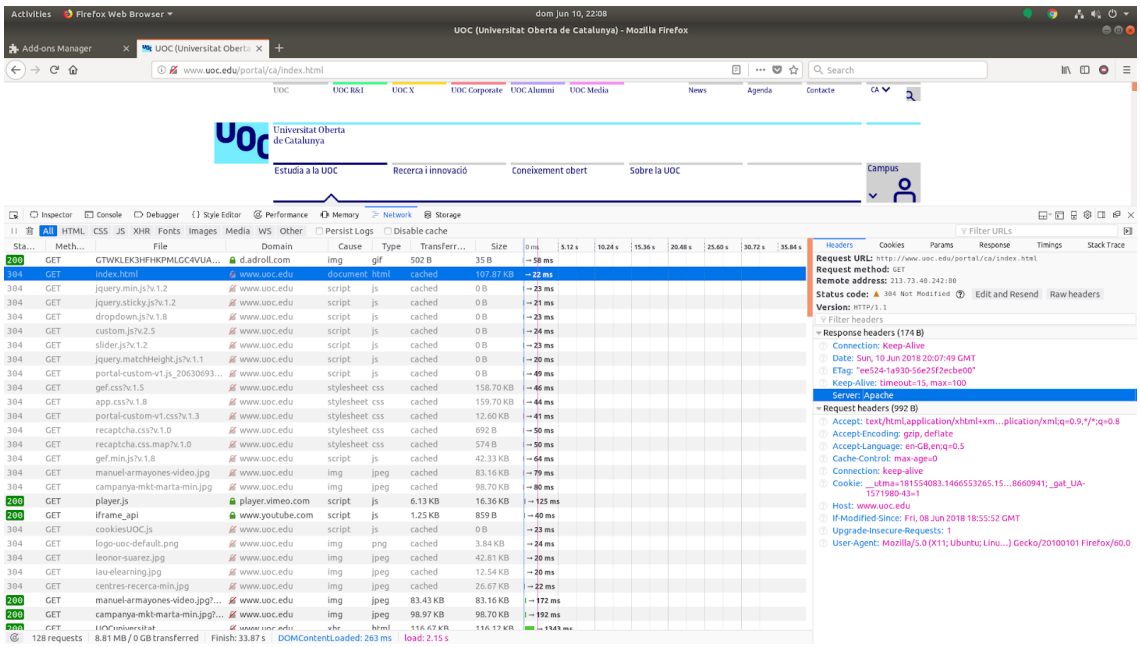


Figura 23: Consola del Firefox per obtenir el header 'server'

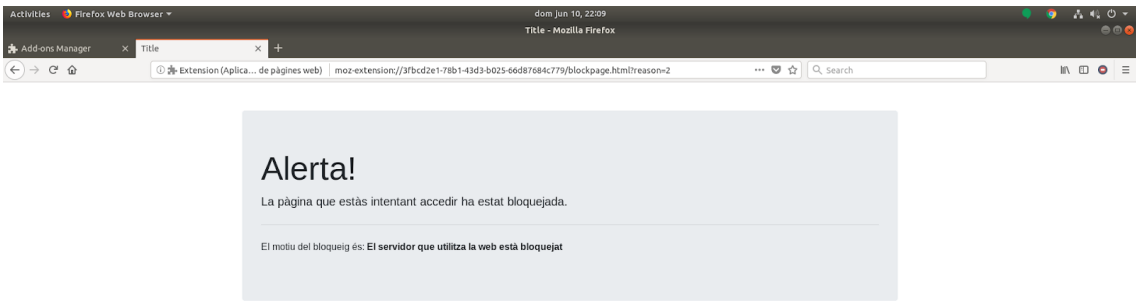


Figura 24: Prova del bloqueig per servidor

- A continuació fem la prova de bloqueig per paraules, posant al camp Bloqueig de paraules la paraula UOC. Al accedir de nou al web de la UOC ens bloqueja l'accés indicant el nou motiu.

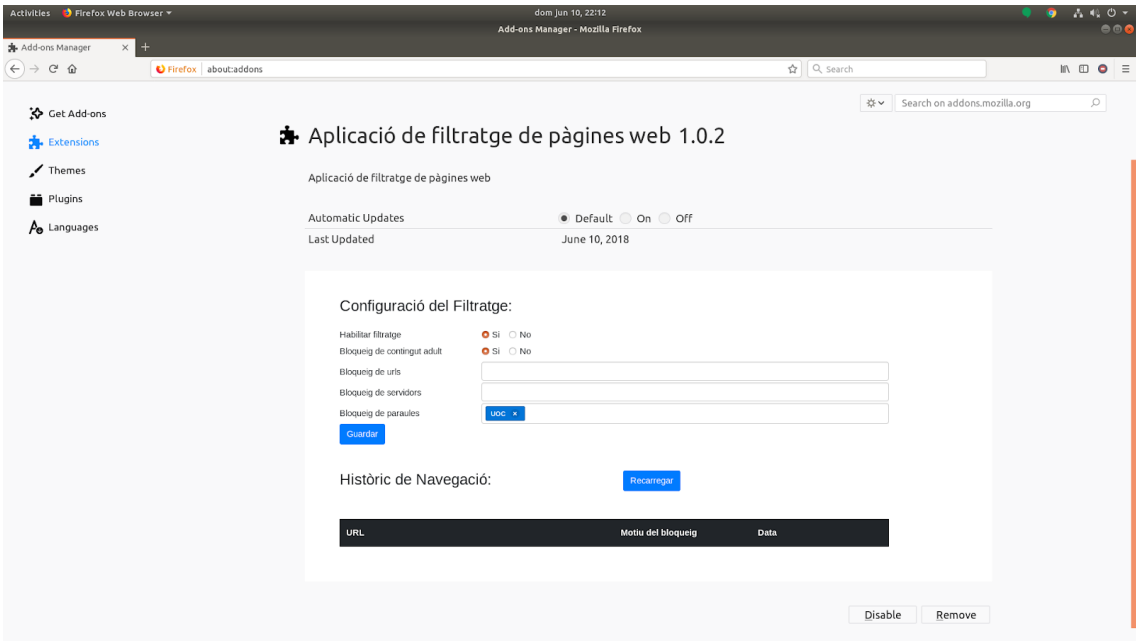


Figura 25: Prova de configuració del bloqueig per paraules

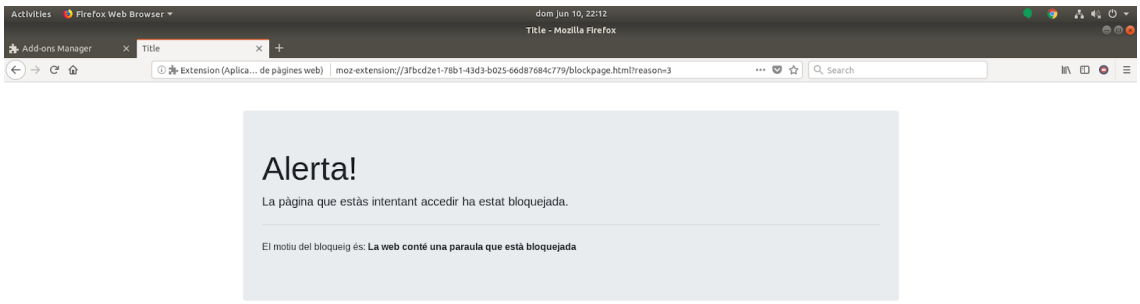


Figura 26: Prova del bloqueig per paraules blocades

- L'última opció de filtratge que ens queda per provar és la de bloquejar webs amb contingut per a adults. Habilitem aquesta opció a la configuració i provem a accedir al web sex.com. Es bloquejarà l'accés a la web indicant que hi ha contingut per a adults.

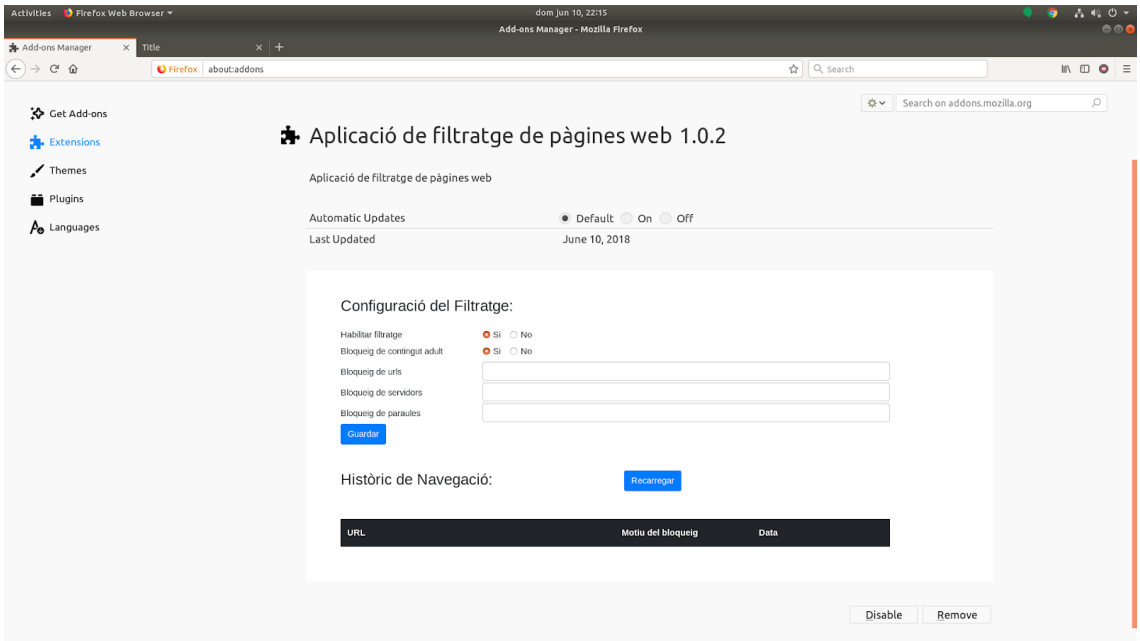


Figura 27: Prova de configuració del bloqueig per contingut destinat a adults

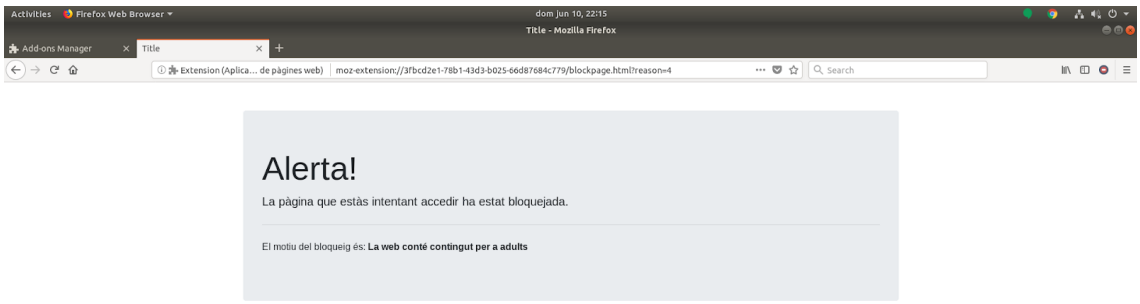


Figura 28: Prova del bloqueig per contingut destinat a adults

- Ara comprovarem si des de la pantalla d'administració es carrega l'historial de navegació correctament. Si fem clic al botó Recarregar ens mostrarà l'historial amb les proves que hem estat fent.

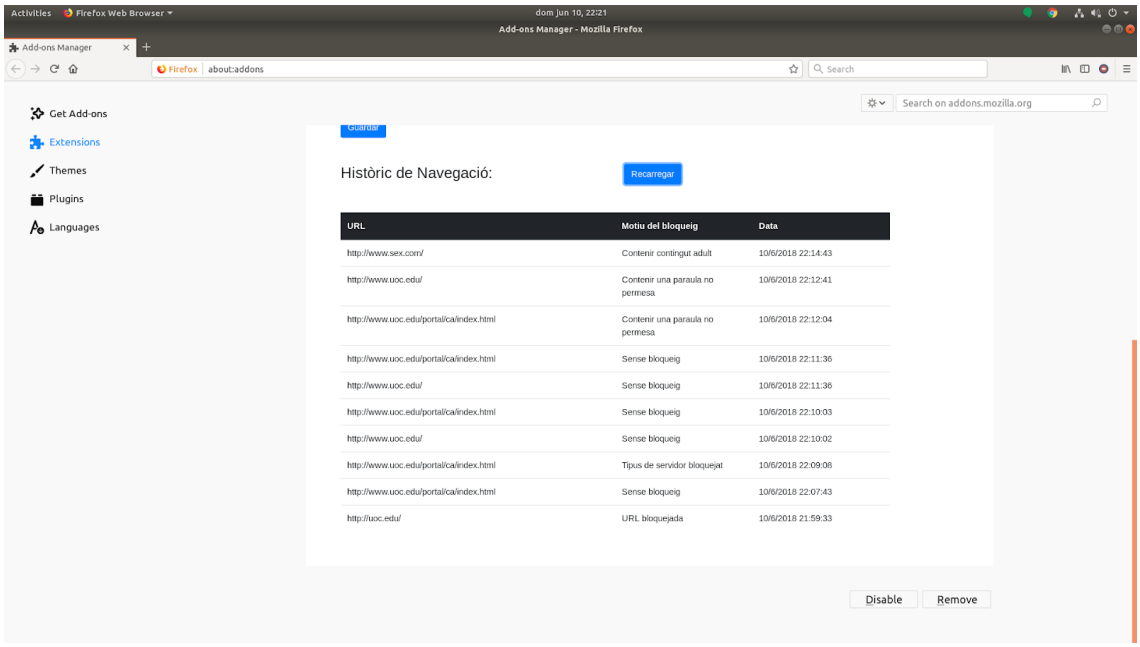


Figura 29: Prova de càrrega de l'historial de navegació

- Hem de comprovar també si quan fem clic al botó de l'extensió ens obre una finestra demanant-nos la contrasenya, per evitar que qualsevol usuari pugui modificar la configuració, i que si la contrasenya és correcte s'obri la pàgina d'administració. Comprovem també que si la contrasenya és incorrecte ens retorna un missatge d'error.

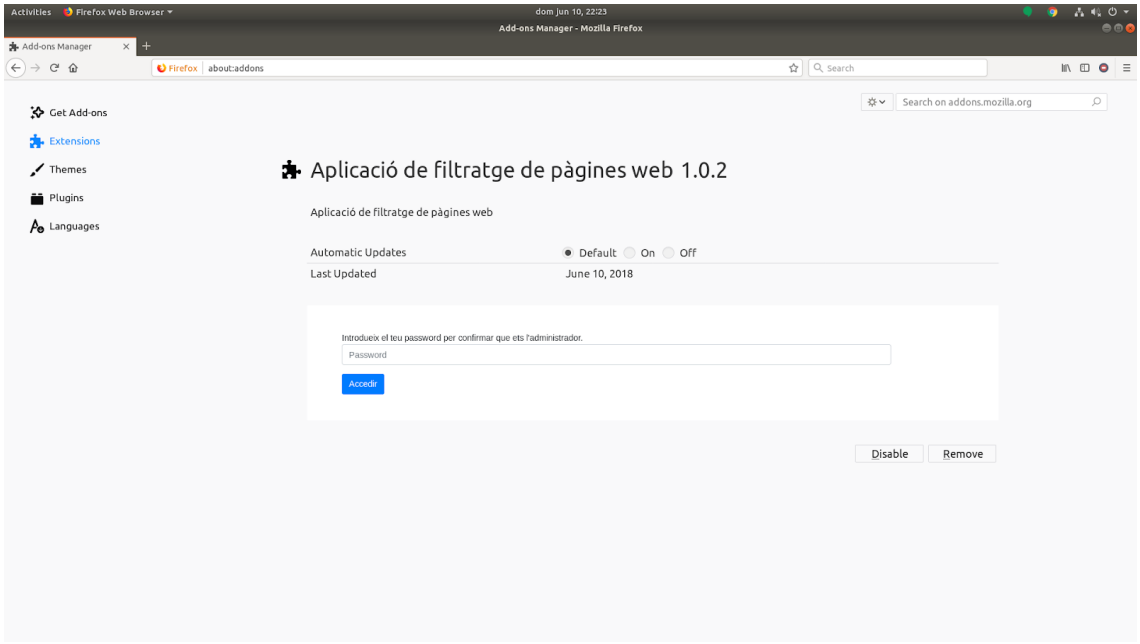


Figura 30: Prova de clic al botó del navegador

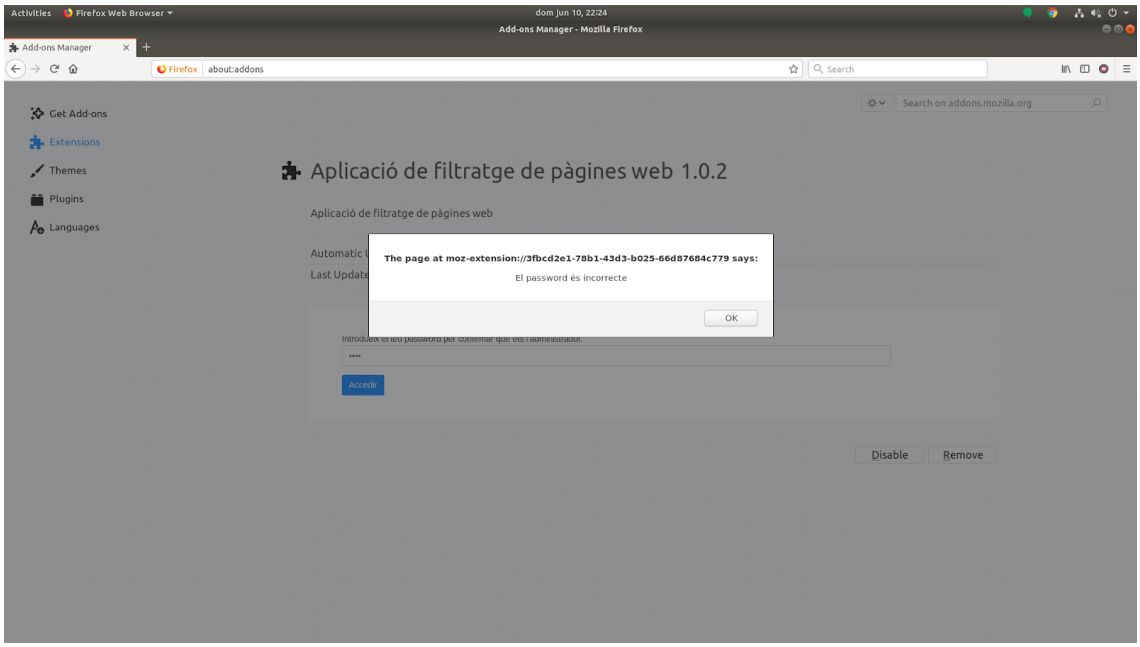


Figura 31: Prova d'inserció de contrasenya incorrecta

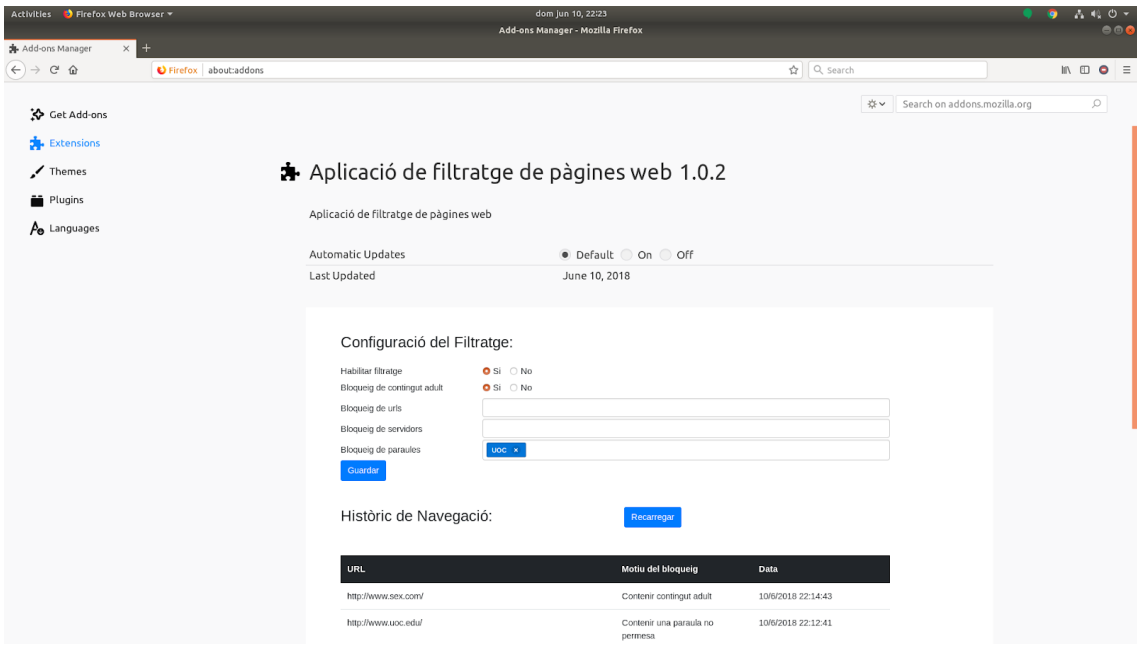


Figura 32: Prova de login amb contrasenya correcta

9. Conclusions

La realització d'aquest projecte m'ha permès aprendre sobre la creació d'una aplicació completa i totalment funcional partint des de zero. La meva experiència prèvia, tant en assignatures prèvies com a la feina, consistia en realitzar només una part del procés, ja sigui l'anàlisi, el disseny o la implementació, i el fet de fer aquest treball m'ha donat una visió més general de com afrontar i planificar tot el cicle de vida d'un projecte.

Per diferents circumstàncies, les primeres setmanes no vaig poder dedicar tot el temps que requeria el projecte, i això va provocar un retràs a la planificació i que hagués de fer un esforç extra al tram final. Això és una cosa que he aprés de cara al futur, he de saber gestionar millor el temps per a conciliar vida laboral, vida familiar i temps d'estudi.

Un inconvenient important que em vaig trobar va ser el problema amb el desenvolupament inicial de l'extensió per al navegador Chrome, que s'exposa a l'apartat 5.1. Això em va fer retardar-me i plantejar-me la viabilitat del tipus d'aplicació escollit, encara que finalment ho vaig poder solucionar amb el canvi a Firefox.

Encara que aquests problemes mencionats no m'han permès seguir el procés de desenvolupament del projecte de forma ideal, crec que he assolit els objectius que em vaig plantejar a l'inici. He pogut desenvolupar una aplicació de filtratge web que compleix els objectius marcats inicialment i que es pot fer servir de manera relativament senzilla.

Una de les millores que es podrien aplicar de cara al futur seria utilitzar blacklists per a millorar el filtratge web. D'aquesta manera es podria bloquejar altre tipus de contingut (anuncis, etc.).

En definitiva, encara que he tingut alguna complicació, em quedo sobretot amb l'experiència obtinguda, que em permetrà gestionar de millor manera la realització de projectes de cara al futur.

10. Glossari

Blacklist: mecanisme bàsic de control que permet l'accés de tots els elements excepte d'aquells que estan inclosos a la llista.

API: acrònim de Application Programing Interface (interfície de programació d'aplicacions), és un conjunt de mètodes i procediments que ofereix una llibreria per a ser utilitzada per un altre software.

API REST: REST és l'acrònim de Representational State Transfer (transferència d'estat representacional). És una interfície entre sistemes que fa servir el protocol HTTP per a obtenir dades o generar operacions.

Proxy: servidor que fa d'intermediari a les peticions de recursos que fa un client a un altre servidor.

Wireframe: es una guia visual que representa l'esquelet o estructura visual d'un lloc web.

11. Bibliografia

https://www.quora.com/What-are-the-ways-to-categorize-a-URL-or-domain-Is-this-URL-likely-to-contain-safe-content-or-is-it-likely-to-have-adult-content

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest

https://developer.mozilla.org/es/Add-ons/WebExtensions/Intercept_HTTP_requests

https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest/StreamFilter

http://www.datumbox.com/machine-learning-api/

https://es.wikipedia.org/wiki/Diagrama_de_flujo

https://es.wikipedia.org/wiki/Wireframe_(dise%C3%B1o_web)

https://nodejs.org/en/

http://expressjs.com/

https://www.mongodb.com/

http://mongoosejs.com/

https://www.npmjs.com/package/bcrypt

https://developer.mozilla.org/es/Add-ons/WebExtensions

https://nodejs.org/es/download/package-manager/#distribuciones-de-linux-basadas-en-debian-y-ubuntu

http://pm2.keymetrics.io/

https://www.digitalocean.com/community/tutorials/como-instalar-nginx-en-ubuntu-16-04-es

https://eladnava.com/binding-nodejs-port-80-using-nginx/

https://www.digitalocean.com/community/tutorials/como-instalar-mongodb-en-ubuntu-16-04-es

https://developer.mozilla.org/en-US/Add-ons/Distribution