Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Herramienta de control parental basada en software libre

Autor: Natalia Sánchez Portillo

Tutor: Fernando Cárdenas Fernández

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Ingeniería de Telecomunicación

Herramienta de control parental basada en software libre

Autor:
Natalia Sánchez Portillo

Tutor:
Fernando Cárdenas Fernández
Profesor asociado

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Herramienta de control parental basada en software libre

Autor: Natalia Sánchez Portillo

Tutor: Fernando Cárdenas Fernández

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis amigos y mis telecos

A esa parte de mí que me empujó a seguir.

Agradecimientos

No ha sido fácil llegar hasta aquí.

Ahora es el momento de echar un vistazo atrás y observar todo el camino recorrido.

No durante toda la trayectoria he podido disfrutar del paisaje tanto como me hubiese gustado. A veces (demasiadas) alguna que otra tormenta me impedía ver lo bonito que estaba el cielo ese día.

Pero qué bonito era cuando todo encajaba, os encontraba y volvía a salir el sol. Al final se trata de eso, ¿no? De coincidir con personas que te hagan ver cosas que tú no consigues ver, que te enseñen a mirar con otros ojos…

Papá, Mamá, Manuel: Por confiar en todo lo que llevaba dentro, cuando ni yo misma era consciente. Por quererme incondicionalmente y por hacerme la niña más feliz del mundo, gracias.

Tita, Tito, Andrea, Rumba: Segunda familia. Segundo hogar. Por apostar y creer en mi siempre. Por hacerme sentir que pase lo que pase, saldré adelante.

A ti, *Juan*: Gracias por aparecer, seguir y quedarte. Por apoyarme, sanarme y aportarme tantas cosas bonitas. Has sido y eres una pieza fundamental en todo esto.

A los que estábais desde el principio y a los que habéis llegado durante el camino para estar: Gracias. Por contagiarme de esa increíble energía y por ayudarme a ser.

A mis profesores y mi tutor, *Fernando*: Por vuestro esfuerzo y confianza. Habéis conseguido que descubra una parte de mi (muy bonita) que ni yo misma conocía.

Sois y seréis el hogar al que siempre regresaré. Lo prometo.

Natalia Sánchez Portillo

Sevilla, 2020

Resumen

El objetivo del presente estudio fue la creación de una herramienta de control parental. Queríamos que fuese exportable a cualquier equipo, por lo que ha sido creada situándonos en el peor de los casos: un usuario que desea utilizar un equipo antiguo, sin capacidad y a simple vista inservible. Para ello, hemos modificado y adaptado una minidistribución Linux de manera que pueda funcionar en el dispositivo gracias a una memoria USB. Al arrancarse de esta manera, las dependencias entre la herramienta y el equipo utilizado serán mínimas.

La minidistribución es ligera. Ocupa el espacio mínimo necesario, ya que sus datos se guardan en la memoria USB. Se ha instalado en ella el Proxy Squid, el encargado de bloquear todas las páginas web que el usuario considere inadecuadas. Además, para garantizar la seguridad de los menores, ha sido incluida una lista recomendada por expertos que indican que son peligrosas y no deben ser utilizadas. Gracias a ella, ninguna persona conectada a esta red podrá acceder a las páginas web incluídas en la lista, a no ser que se elimine de la misma.

Para que el usuario pueda bloquear las páginas web de una manera sencilla y rápida, se ha desarrollado una aplicación web. Dicha aplicación sólo será accesible por los usuarios registrados en ella. Desde la misma, los usuarios pueden bloquear, desbloquear y listar las páginas web bloqueadas por ellos. Para facilitar la tarea, la propia aplicación recomienda algunas páginas web que se consideran no aptas para los menores y los usuarios también tienen la posibilidad de añadirlas en su lista de páginas bloqueadas.

Dicha aplicación ha sido encapsulada y desplegada también en nuestra distribución gracias a los servicios que nos ofrece Docker y su nube.

Abstract

The objective of the present study was the creation of a Parental control tool. We wanted it to be exportable to any computer, therefore it has been created situating ourselves in the worst possible situation: an user wishing to use a useless old computer which lacks capacity. In order for that to be possible, we have modified and adapted a Linux distribution so that it can work in the computer thanks to a USB memory. By starting up in this concrete form, the dependencies between the tool and the computer will be minimum.

The distribution is lightweight. It takes the minimum space needed, since its data is collected in the USB Memory. Proxy Squid, which is in charge of blocking every webpage considered inadequate by the user, has been installed in it. Furthermore, to guarantee underage people's security, a list with unsafe websites recommended by experts has been included. It is thanks to it that any individual connected to the network will be unable to Access the websites included on the list, unless they are erased.

For the user to be able to block websites in a fast and easy way, a web application has been developed. The mentioned application will only be accessible by registered users. Users may block, unblock and create lists of blocked sites. To make it easier, the application itself recommends some websites which could be considered inadequate for minors and gives users the opportunity to add them to their individual list.

The mentioned application has been encapsulated and displayed in our distribution thanks to the services offered to us by Docker and its cloud.

Índice

ÍNDICE DE FIGURAS

Notación

USB	Universal Serial Bus
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Location
IETF	Internet Engineering Task Force
CA	Autoridad de Certificación
ITU	Unión Internacional de Comunicaciones
ANCERT	Agencia Notarial de Certificación
FNMT	Fábrica Nacional de Moneda y Timbre
BBDD	Bases de datos
RRSS	Redes Sociales

1 INTRODUCCIÓN

Sé que parece que el mundo se está desmoronando, pero en realidad es una gran época para volvernos un poco locos, seguir nuestra curiosidad y ser ambiciosos. No abandonéis vuestros sueños. ¡El mundo os necesita!

-Jeff Bezos-

Hoy en día, las nuevas tecnologías están tomando un gran protagonismo en nuestra sociedad. Forman parte de nuestras vidas, al fin y al cabo, son uno de los pilares principales de la comunicación. No sólo agilizan y perfeccionan muchas de nuestras tareas, si no que también están presentes en nuestras actividades de ocio. Pero, ¿hasta qué punto conocemos las virtudes y los peligros de las tecnologías que nos rodean?

1.1 Motivación

1.1.1 Motivación personal

¿Quién no ha entrado en un restaurante y no se ha encontrado la imagen de una familia sentada en la mesa, con los más pequeños utilizando algún dispositivo electrónico mientras esperan su comida? O incluso peor: mientras comen. Probablemente hace unos años esa imagen fuese impensable. Sin embargo, en la actualidad la gran mayoría de los niños saben qué es una tablet, un móvil o un portatil.

Nacen rodeados de dispositivos. Crecen viendo como sus padres, tíos e incluso abuelos, los utilizan para un sin fin de tareas. Teletrabajo, redes sociales o por qué no para jugar a algún que otro juego. Muchas de estas tecnologías se han convertido en el día a día de las familias para entretener y/o educar con juegos interactivos a los pequeños de la casa. Y ellos, lo normalizan hasta el punto que ellos también se ven capaces de manejarlos.

La capacidad de autoaprendizaje de los niños con el uso de internet y las nuevas tecnologías es sorprendente, comenzando su vida digital cada vez antes. Según datos de UNIR, “uno de cada tres niños pasa de media tres horas diarias conectado a internet”. Pero, ¿qué páginas visita? ¿Qué aplicaciones utiliza? ¿Son apropiadas para su edad? ¿Serán seguras? Esta situación hace que actualmente las familias tengan la responsabilidad de gestionar el uso de los dispositivos. De esta forma, el contenido y las aplicaciones que el niño use podrán ser supervisadas y controladas.

Pero, ¿cuál es la forma más sencilla, eficiente y rápida de hacerlo sin que los más pequeños lo noten? Utilizaremos una **herramienta de control parental**.

1.1.2 Motivación técnica

En primer lugar, es importante destacar la importancia de este proyecto. ¿Quién no ha pensado que está guardando un ordenador antiguo y que no va a ser útil nunca más? Ya sea por la falta de capacidad del disco, por la lentitud del dispositivo o por la propia antigüedad del mismo. Con el desarrollo de este proyecto, conseguiremos **reutilizar un equipo** de manera que pueda actuar de proxy y sea capaz de bloquear todas las páginas web inapropiadas. Son muchas las distribuciones disponibles que pueden ejercer esta tarea, pero mediante nuestra investigación, **adaptaremos una minidistribución Linux** añadiéndole las funcionalidades que sean necesarias.

Por otra parte, no podemos olvidar la gran importancia de las aplicaciones web en nuestro dia a día. Aprenderemos a **desarrollar una aplicación web, protegida y sencilla** de manera que sea útil para cualquier usuario. La protección de los datos se verá reflejada en que únicamente podrán acceder al sistema aquellas personas registradas en la propia aplicación, evitando la manipulación de los mismos y **garantizando la seguridad de la red**. Además, la aplicación contará con su propio certificado autofirmado y así poder usar protocolos seguros para ofrecer más seguridad a sus usuarios.

1.2 Objetivos

El objetivo principal de este TFG consiste en el desarrollo de una herramienta de control parental fácil de usar. Su desarrollo se divide en cuatro partes diferentes:

1) En primer lugar, **la investigación e implementación de una minidistribución Linux** de poca capacidad.
   - Nos permitirá conocer las minidistribuciones Linux ya existentes en la red y obtener las capacidades necesarias para poder adaptarlas a nuestras necesidades.
   - Al ser de poca capacidad, podremos salvar equipos viejos en desuso por sus pocos recursos.
   - Reutilizaremos un **equipo antiguo** instalándole dicha distro. Será personalizada y desarrollada sin olvidar que queremos que ocupe poco espacio en el disco**. Actuará de servidor** gracias a todas las herramientas que dispondrá.
2) En segundo lugar, **el estudio de Spring Framework y el desarrollo de una aplicación web** completo.
   - Estudio de la instalación y configuración de Spring: Cómo utilizar Spring desde cero, aprendiendo a añadir las dependencias y módulos necesarios.
   - Estudio del patrón MVC aplicado a Spring.
   - Estudio de la seguridad en Spring: Utilizaremos "Spring Security", módulo que nos permitirá autenticar usuarios y bloquear accesos.
   - Generación de autocertificados para que la aplicación cuente con protocolos seguros (SSL) e implementar una barrera más de seguridad.
3) En tercer lugar, utilizaremos el servicio que nos ofrece **Docker** para poder instalar con facilidad el servidor en nuestra minidistribución.
4) En cuarto lugar, la investigación y uso del **proxy web SQUID**.
   - Nos restringirá el acceso a las páginas web que elijamos.

Gracias a todas estas herramientas, podremos **controlar la navegación por Internet** de los menores.

2 ANÁLISIS DEL PROBLEMA

La necesidad de crear esta herramienta nace del miedo. Internet se ha convertido en una herramienta fundamental en la vida de muchas personas, y sorprendentemente, también para los más pequeños. Sin embargo, también entraña riesgos.

2.1 Definición del problema

Podríamos enumerar una gran lista sobre todos los peligros que existen en Internet para los menores:

- Acceso a información inadecuada para su edad
- Acceso a información peligrosa
- Anuncios falsos
- Riesgos del uso imprudente de las redes sociales

Todas las páginas que ofrecen información de este tipo son públicas y accesibles desde cualquier dispositivo. Basta con introducir en el navegador qué buscamos, y en un unos segundos, nos mostrará miles de sitios donde encontrarlo. Esto hace que sea imposible controlar qué páginas se visita con la configuración por defecto que traen de fábrica, ya que están disponibles para todo el mundo.

2.2 Solución propuesta

Por lo mencionado anteriormente, proponemos una solución:

Gracias a la herramienta de control parental, los usuarios de este servicio podrán bloquear el acceso a cualquier página web dentro de su red.

De esta manera, los menores quedarán menos expuestos a los peligros que trae consigo Internet.

3 BASE TEÓRICA

Si no vas hasta el final, ¿por qué empezar?.

- Joe Namath -

Para asegurar el entendimiento de cómo funciona nuestro servicio, previamente explicaremos una serie de conceptos y el contexto en el que nos encontramos.

3.1 Descripción del escenario y conceptos implicados

Supongamos que vivimos en una casa con un pequeño de tan solo 7 años de edad (podemos imaginar lo curioso y las ganas que tiene el menor de aprender y conocer). Después de leer algunos estudios, llegamos a la conclusión de que es bueno para su crecimiento y aprendizaje que nazca usando las tecnologías poco a poco, así que a ciertas horas del día usa un ordenador. Quiere que sepa que confías en él, así que le deja "solo". ¿Por qué "solo"? Porque estará activada la herramienta de control parental.

Lo que el pequeño no sabe es que, en otra habitación, cuenta con un equipo muy antiguo (ni sabe de su existencia) actuando de herramienta de filtrado de contenido. En él, estará instalada la minidistribución Linux elegida, con nuestra aplicación web desplegada (gracias a Docker y Spring, de los que hablaremos más adelante) y con un proxy web activado (encargado de bloquear cualquier **URL**). De esta manera, podremos controlar la navegación en Internet de todos los dispositivos que se encuentren conectado al **proxy**.

Para resolver el escenario descrito en el apartado anterior, nuestra idea consiste en crear una aplicación web sencilla, práctica y fácil de usar, que junto al proxy nos permita bloquear el acceso a ciertas páginas web.

3.1.1 URLs

Para controlar el acceso a ciertas páginas webs, lo que realmente estamos haciendo es bloquear su **URL**.

Cuando hablamos de URLs (Uniform Resource Location) hacemos referencia a una serie de caracteres que permite localizar ciertos recursos en Internet. Cada vez que se sube a la red algún fichero se le asigna su propia URL, de forma que quedará identificado univócamente. Con otras palabras, podemos referirnos a esta identificación como su dirección dentro de Internet.

El esquema de la URL también nos indica el protocolo utilizado para obtener la información del recurso solicitado. En el caso del ejemplo mencionado, sería HTTP (HyperText Transport Protocol), del que también hablaremos a continuación.

Por tanto, para restringir el acceso a ciertas páginas web, únicamente tendremos que indicar el nombre de su URL para bloquearlas. En la siguiente figura podemos observar un esquema de la estructura de una URL:

Figura 1: Esquema de la estructura de una URL

3.1.2 Proxy

Cuando navegamos por Internet y queremos ver una página web específica simplemente la seleccionamos y nos aparece su contenido. Sin embargo, no nos planteamos qué pasos intermedios se ejecutan hasta llegar ahí. Pues bien, en el caso de configurar y activar un proxy, este será quien se comunique con el sitio web en cuestión.

Con otras palabras, un **proxy** no es más que un **dispositivo intermediario** entre el usuario e Internet. Todas las consultas que el usuario haga en Internet, antes las capturará el proxy y las analizará. Gráficamente podríamos representarlo con la siguiente imagen:

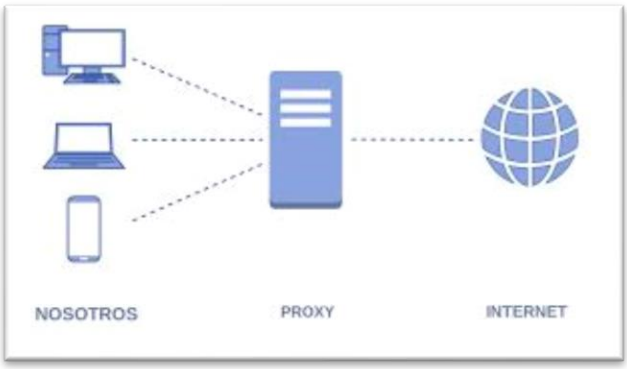

Figura 2: Representación proxy

Si tomamos como ejemplo la **Figura 2**, nos imaginamos que queremos visitar cierta página web que se encuentra en Internet. Si no tuviésemos activado ningún proxy, directamente Internet nos devolvería la información solicitada. Sin embargo, al tener configurado y activado nuestro proxy, la solicitud antes de llegar a Internet la interceptará el dispositivo. Una vez que el proxy haya revisado su propia configuración, aprobará o rechazará la petición. En el caso de tener permitido el acceso, el proxy recibirá la información de Internet y nos responderá con el contenido de la página web. Si por el contrario la rechaza, nos enviará un mensaje de error. Por tanto, podemos afirmar que no existe una conexión directa entre "nosotros" e "Internet", si no que antes tendrá que pasar por el proxy. También es importante destacar que "Internet" es ajeno a que el proxy actúa de intermediario.

En Internet un dispositivo se suele definir como "cliente" o "servidor" en función del papel que tenga. En nuestro escenario, el cliente será el navegador que utilice el usuario. Los servidores serán los que procesen las solicitudes de los clientes. Este intercambio de información será lo que denominemos como tráfico de la red.

En el caso de no tener proxy, las solicitudes las tramita directamente el servidor. Pero en nuestro caso, será el proxy quien se encargue de esta tarea. Gestionará el tráfico, ya que el cliente hablará solo con el proxy.

Cuando Internet le responda al proxy con la información solicitada por el cliente, este le enviará el contenido.

Dentro de este tipo de escenario, podemos clasificar a los servidores proxys como local o externo:

- Local: El servidor proxy se encuentra en el mismo equipo desde donde se realiza la petición a Internet. Este tipo se suele utilizar cuando es el propio usuario el que realiza los filtros
- **Externo:** En este caso se encuentra en un equipo diferente. Se utiliza para el control y bloqueo de tráfico. Este tipo será el que se utilice en el proyecto.

Una vez entendido el concepto de proxy, es momento de plantearnos para qué sirve exactamente. Según [1] y [2] son muchas las funciones que puede ofrecer un proxy: control de acceso, registro de tráfico, mejora del rendimiento… Vamos a detallar algunas de ellas:

- **Controlar el acceso a Internet:** Al ser el proxy quién gestiona las solicitudes del cliente, decide a qué páginas web permite el acceso y cuáles no (en función de lo que esté configurado).
- **Registra el tráfico:** Sabe quién se conectó, cuándo y por dónde navegó. Puede ser muy útil para equipos corporativos e impresas interesadas en saber qué páginas visitaron sus empleados.
- **Mejora el rendimiento:** El proxy posee una memoria caché donde guarda algunos recursos. Si algunos de ellos son solicitados por segunda vez, no tiene que consultarlos en Internet, si no que directamente se lo ofrece. Agiliza el proceso.
- **Anonimato:** Como actúa de intermediario, Internet no conoce a su "verdadero cliente". Mejora la seguridad.

También podemos encontrarnos con algunas desventajas:

- Si muchos ordenadores hacen uso del mismo proxy, puede verse sobrecargado y tardar en procesar sus solicitudes.
- Al guardar cierta información en su caché puede incluso guardar copias de sus transferencias, reduciendo en este caso la seguridad de los datos.
- Debido a errores de la caché también puede ocurrir que se encuentre desactualizado. Es decir, que la información que le envie al cliente no sea la última.

Hoy en día existen diferentes tipos de proxy dependiendo de su uso principal:

- **Proxy Web**: Se encarga de gestionar los protocolos HTTP, HTTPs y FTP. Se podrán realizar filtrados en función de su URL. Gracias a ellos se podrá mejorar la seguridad y el anonimato, reducir el tráfico y agilizar la velocidad de navegación. Será el que se utilice en el proyecto.
- Proxy NAT: Su principal función será traducir las direcciones privadas en direcciones públicas para que puedan alcanzar su destino. La traducción de direcciones también se denomina enmascaramiento de IPs. El proxy se encarga de reescribir estas direcciones IPs para realizar las peticiones de los clientes y a su vez ofrecer la información del servidor al cliente. Serán necesarios en el caso de hogares y empresas con varios equipos en red y con acceso a Internet.
- Proxy Inverso: Es un proxy que se encuentra en uno o varios servidores web. Recibirá todas las peticiones que tengan como destino alguno de los servidores web. Fortalecerá la seguridad de los servidores. Además, podrá distribuir la carga entre varios servidores, agilizando el proceso.

- Proxy transparente: No necesita que el navegador (cliente) esté configurado, por lo que el usuario no sabrá que se encuentra conectado al proxy y por tanto no sabrá que sus peticiones son gestionadas por este dispositivo antes de llegar a Internet. Suele ser utilizado en los equipos corporativos de las empresas.
- Proxy abierto: Acepta cualquier petición, aunque venga de un equipo que no se encuentre conectado a su red.

Nosotros lo utilizaremos **para la restricción de determinadas URLs** y para ello utilizaremos un proxy web. Al tratar tráfico HTTP/HTTPs su arquitectura será la siguiente:

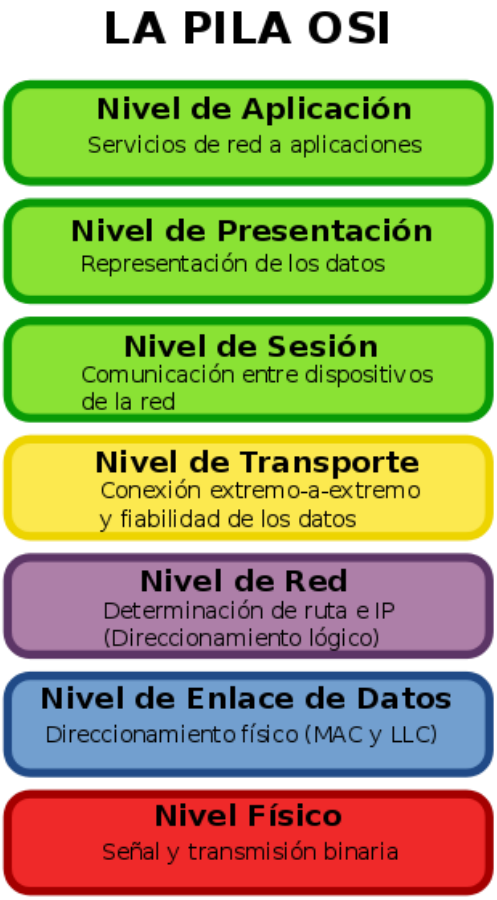

Figura 3: Arquitectura proxy web

El proxy que usaremos será **SQUID,** y en los próximos apartados lo explicaremos con más detalle.

3.2 Protocolos utilizados por el servidor web

Como se ha explicado en los apartados anteriores, se ha desarrollado una **aplicación web** para que el Usuario pueda **bloquear y desbloquear las páginas web** que quiere que sean o no accesibles. La aplicación se ha desarrollado a través de **Spring Boot**, el cual se encarga de configurar un servidor web, **Tomcat**.

Un servidor web (como es el caso de Tomcat) se encarga principalmente de ofrecer determinada información hacia el exterior. Esto lo hará cuando el cliente, mediante una petición HTTP, solicite ciertos datos. Este proceso se puede representar en la siguiente figura:

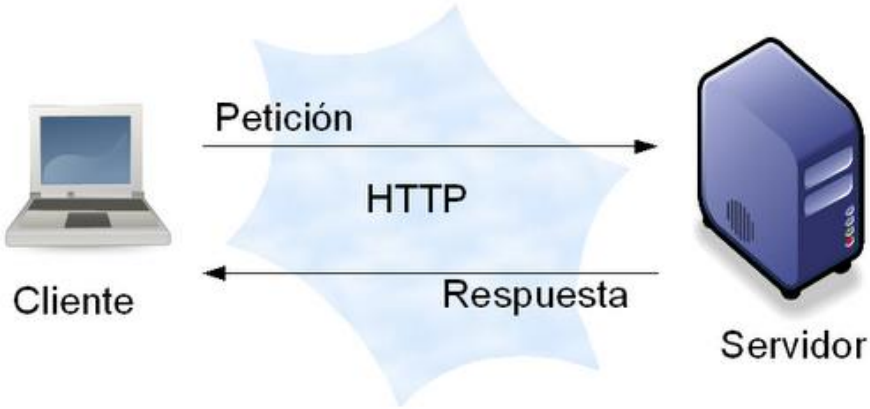

Figura 4: Relación entre Cliente y Servidor

El cliente puede enviar peticiones HTTP (GET, POST…), y el servidor, una vez procese la petición, contestará. Por ejemplo: el cliente solicita la página web (al servidor le llega una petición GET) y el servidor le enviará su contenido. Otro ejemplo podría ser que para que el cliente pudiese acceder a la página web, tuviese que introducir su usuario y contraseña en un formulario (petición POST). En ese caso, l servidor comprobará si son datos correctos y si es así, el cliente podrá ver la pantalla de menú y ya hacer sus correspondientes tareas.

Para poder gestionar correctamente un Servidor Web, previamente es importante conocer su arquitectura y los protocolos implicados. Se han utilizado [4] y [5] como referencia. En la siguiente figura vemos la torre de protocolos de un servidor web:

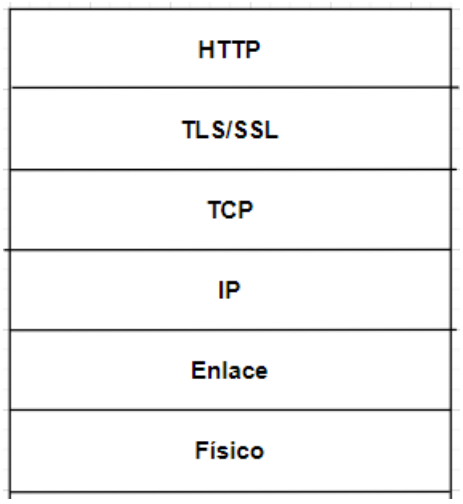

Figura 5: Torre de protocolos servidor web

3.2.1 Protocolo web HTTP

HyperText Transfer Protocol (HTTP), según la IETF, es un protocolo de comunicación que se utiliza entre los navegadores y servidores web para transferir información entre sí. Su uso implica normalmente otros componentes web como el lenguaje de etiquetas HTML o descriptores de dirección URL.

En la siguiente figura podemos observar la estructura de una petición HTTP:

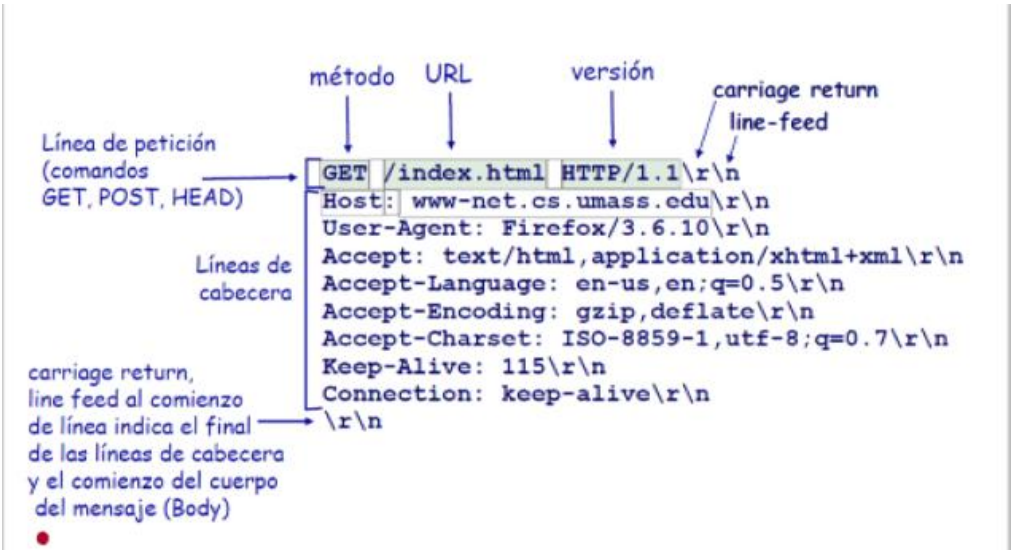

Figura 6: Estructura petición HTTP

3.2.2 Protocolo web HTTPs

Para reforzar la seguridad de la aplicación, los servidores web pueden ofrecer algunos mecanismos de seguridad mediante el protocolo SSL (Secure Socket Layer) o TLS (Transport Layer Security). Ambos se basan en la utilización de certificados digitales.

El usar HTTP sobre TLS/SSL define el protocolo seguro HTTPs

3.2.3 Encriptación

Cuando el navegador y el servidor web intercambian información se encuentra cifrada. La encriptación de la información se clasifica en función de la clave empleada:

- Simétrica: En este caso los dos extremos de la comunicación usan la misma clave. Por tanto, se utiliza tanto para cifrar como para descifrar.

  Algoritmos que utilizan este tipo de cifrado: IDEA-ECV

- Asimétrica: Lo característico de este tipo de encriptación es que existen dos claves, una privada y otra pública. La información que se cifra con la clave pública solo se puede descifrar con la privada y al revés.

  Algoritmos que utilizan este tipo de cifrado: DH/DSS

También es posible combinar ambos métodos, denominado cifrado híbrido. Los protocolos SSL/TLs utilizan este tipo de cifrado.

3.2.4 Certificación

En el caso de HTTPs, los servidores web se identifican a los navegadores utilizando un certificado. Existen dos tipos de certificados:

- Certificados generados por una Autoridad de Certificación (CA): Una entidad externa asegura la identidad del propietario del certificado.
- Certificados autogenerados: Ninguna CA garantiza la identidad. Son generados por el propio servidor. Se utilizará este tipo de certificado en el proyecto.

El formato estándar de los certificados digitales es el X.509 (recomendato por la ITU). Cuando se genera el certificado, el siguiente paso es instalarlo en el servidor.

3.2.5 Protocolo TLS/SSL

Gracias a este protocolo se garantiza:

- Autenticación del servidor: Antes de la transmisión de información, se comprueba que el certificado es válido.
- Confidencialidad: Gracias al certificado digital se consigue la identidad del servidor web.
- Integridad de los datos recibidos: Se comprueba que los datos han llegado en su totalidad.

HTTPs sobre SSL normalmente tiene asignado el puerto 443 y no el puerto 80 (como en el caso de HTTP a secas). Cuando queramos acceder a páginas web seguras: Esto se indicará directamente sobre la dirección de la URL: https://direccion:443

3.2.6 Protocolo TCP

Protocolo de Control de Transmisión.: Este protocolo se encarga de crear “conexiones” entre sí para que se cree un flujo de datos. Este proceso garantiza que los datos sean entregados en destino sin errores y en el mismo orden en el que salieron. También se utiliza para distinguir diferentes aplicaciones en un mismo dispositivo.

3.2.7 Protocolo IP

Protocolo de Internet: Este protocolo se encarga del envío y recepción de datos en bloques. El envío lo hace siempre por la mejor ruta pero sin garantizar que llegue a alcanzar el destino.

4 ESTADO DEL ARTE

Cada vez son más jóvenes los menores que empiezan a descubrir el gran mundo que ofrece Internet, principalmente haciendo uso de teléfonos móviles o tablets. Por eso, es de vital importancia tanto una buena educación digital, como la instalación y uso de algunas herramientas de control parental que sean capaces de protegerlos de todos los peligros que rondan por la red. Mencionaremos algunos de ellos.

4.1 Ejemplos de herramientas ye existentes

4.1.1 Qustodio

Se trata de una de las herramientas de control parental más recomendadas y conocidas. Se encuentra disponible tanto para ordenador como para dispositivos móviles.

Figura 7: Logo de la aplicación QUSTODIO

Ofrece una versión gratuita y otra de pago. Algunas de las funciones premium son:

- **Informes detallados**

  Podrás recibir en tu bandeja de entrada un documento donde quede reflejado toda la actividad de en la red de forma diario o semanal.

- **Bloqueo de pornografía**

  Se encargará de bloquear cualquier contenido inadecuado gracias a sus filtros y restricciones.

- **Posibilidad de decidir cuánto tiempo están conectados**

  Para evitar que los más pequeños pasen demasiadas horas frente a la pantalla, la aplicación permite restringir las horas de acceso a la red

- **Bloqueo de juegos y aplicaciones**

  Qué aplicaciones y juegos estarán permitidos en tu dispositivo.

- **Monitorización de youtube**

  Permitirá la consulta del historial de youtube junto con sus reproducciones.

- **Seguimiento en redes sociales**

  Facebook o Instagram entre ellas

- **Posibilidad de controlar llamadas y SMS en Android**

  Registrará con quién hablan e intercambian mensajes,

- **Localizador familiar**

Al guardarse la localización del dispositivo, se podrá controlar dónde han estado los usuarios.

- **Botón de pánico en Android**

  Si los usuarios se encuentran en peligro tienen esta opción para mandar un mensaje SOS con su posición exacta.

Podrá encontrar más información acerca de esta herramienta en [6].

4.1.2 Secure Kids

Secure Kids es una de las aplicaciones más completas del mercado. Gracias a su sencilla interfaz gráfica, es fácil de usar y ofrece protección tanto a padres como a niños. Con su sistema de geolocalización avanzado, mantendrá al usuario del dispositivo localizado en todo momento. Además su sistema de emergencia notificará en cada situación peligrosa.

Figura 8: Logo de la aplicación SECUREKIDS

Ofrece una versión gratuita con multitud de opciones, en las que se encuentra:

- **Bloqueo de aplicaciones**

  Activando el módulo “Bloqueo de aplicaciones” podrás restringir el uso de ciertas páginas web sin ninguna dificultad.

- **Descansos**

  Planificar descansos después de usar el dispositivo varias horas seguidas es muy importante para la salud del usuario, por ello, gracias a este módulo se permitirá programar pausas durante el tiempo que se desee.

- **Botón de emergencia**

  Gracias a este módulo, al pulsar en el botón de emergencia se hallará la localización del dispositivo en ese momento. Así asegura la tranquilidad de los familiares.

- **Configuración remota**

  Permite a los administradores de la aplicación gestionar su uso desde cualquier otro dispositivo, sin distinguir entre tablets o teléfonos móviles.

- **Geolocalización**

  Nació del miedo de los familiares de dejar solos a los más pequeños. Permitirá conocer la localización del dispositivo.

- **Bloquear llamadas**

Gestiona los usuarios permitidos y no permitidos a la hora de realizar una llamada

- **Control de páginas web**

  Permite restringir el acceso a cualquier página web pública.

- **Crear alarmas**

  Gracias a esta opción se podrán planificar alarmas para cualquier momento

- **Seguridad online**

  Ofrece seguridad

Podrá encontrar más información acerca de esta herramienta en [7].

4.1.3 Eset Parental Control

Se trata de una aplicación para dispositivos Android. Permite gestionar la actividad en la red de los más pequeños y garantizar su seguridad.

Figura 9: Logo de la aplicación ESET

Es otra de las herramientas de control parental que ofrece versión gratuita y premium.

Alguna de las opciones que contiene la versión gratuita son:

- **Control de aplicaciones**

  Cuenta con filtros dependiendo de la edad de los menores para gestionar qué aplicaciones están permitidas y cuáles no.

- **Límite de tiempo para juegos**

  Permite configurar el tiempo máximo de uso de ciertos juegos.

- **Informes básicos**

  Ofrece cierta información básica sobre el uso que se le ha dado al dispositivo

Sin embargo, la versión premium además ofrece entre ellas:

- **Control web y Filtro de búsqueda segura**

  Gracias a sus filtros, se podrá gestionar qué recursos web están accesibles y cuáles no.

- **Geofencing**

Es una tecnología que permite, gracias a la localización del dispositivo, establecer "vallas virtuales". En otras palabras, delimita zonas geográficas reales. Esto permite mandar mensajes en función de dónde se encuentre, recopilar información de la ubicación…

- **Localizador del niño**

  Registra la posición exacta del dispositivo, por lo que se podrá controlar dónde se encuentra en todo momento.

- **Informes completos**

  Ofrece información más avanzada y completo sobre el uso del dispositivo.

- **Todas las opciones gratuitas**

  Contiene las opciones mencionadas anteriormente.

Para hacer uso de esta herramienta, son de obligatorio cumplimiento los siguientes requisitos:

- Android 4.0 y posteriores
- Memoria RAM interna 512 MB o más
- Procesador: ARM Core
- Arquitectura: ARMv7 o posterior, mínimo 500 MHz
- Pantalla táctil (mínimo 320x480px)
- Conexión a Internet
- Servicios de Google Play y Accesibilidad

Podrá encontrar más información acerca de esta herramienta en [8].

4.2 Nuestra propia herramienta de control parental

¿Por qué crear una nueva existiendo actualmente diversas herramientas de control parental?

En primer lugar por uno de los objetivos principales del proyecto: **conseguir reutilizar equipos antiguos** y aparentemente inservibles. Todas estas herramientas mencionadas necesitan un hardware y software específico que no cumpliría un equipo con más de 10 años. Al ser una herramienta personalizada, se ha podido adaptar a nuestras necesidades.

Todas las mencionadas están orientadas al uso principalmente en teléfonos móviles o tablets. Estos tipos de dispositivos se usan frecuentemente fuera del domicilio familiar. Por ello, casi todas las herramientas mencionadas cuentan con geolocalizador y algunas utilidades para ello. Sin embargo, nuestra herramienta al estar orientada para PCs de sobremesa o portátiles muchas de esas opciones son innecesarias. La aplicación personalizada posee lo justo y necesario para no estar "sobrecargada".

5 ENTORNO DE TRABAJO Y TECNOLOGÍAS

Muchas veces la gente no sabe lo que quiere hasta que se lo enseñas.

-Steve Jobs-

Antes de decidir entre una tecnología u otra, fue necesario un estudio previo de las diferentes opciones existentes. De esta manera, se pudo elegir la óptima y la que se asemejaba más al fin que estábamos buscando. Explicaremos de forma detallada los diferentes sistemas.

5.1 Conceptos básicos sobre minidistribuciones

Para entender el concepto de minidistribución, en primer lugar tenemos que conocer **qué es un sistema operativo**. Un sistema operativo es un conjunto de programas que sirven para la gestion de nuestro dispositivo. Hará de interfaz entre el sistema y el usuario. Entre todas las tareas que administra nos encontramos:

- **Administración del procesador:** Mediante un algoritmo de programación, se encarga de la distribución del procesador entre los diferentes programas.
- **Gestión de la memoria:** Es el encargado de asignar cierta memoria a cada programa. Si no hubiese suficiente espacio libre en la memoria física, también es capaz de crear "memorias virtuales" para ejecutar ciertas aplicaciones.
- **Manejo de los periféricos (entradas/salidas):** El sistema operativo también se utiliza para gestionar las interacciones con el hardware. Mediante sus drivers (administradores de los periféricos) controlará el acceso de los programas a ciertos recursos. Compartirá los recursos hardware entre los usuarios.
- **Gestión de ejecución de aplicaciones:** Se asegura de que las aplicaciones puedan ejecutarse con normalidad asignándole los recursos necesarios.
- **Administración de permisos:** Cada Usuario será propietario de ciertos permisos y será el sistema operativo el encargado de asignarlos, gestionarlos y aplicarlos.
- **Gestión de archivos:** Una vez asignados los permisos a cada Usuario, el sistema operativo organizará la lectura y escritura en los diferentes archivos.

Entonces, **¿qué es una minidistribución?**

Según [9] una minidistribución no es más que una variante del sistema operativo, con el objetivo de incoporar ese sistema en alguna unidad de almacenamiento como un disco o usb. Nos centraremos en las minidistribuciones linux ya que nuestro objetivo es trabajar con software libre.

Gracias a este tipo de distribuciones, podremos encender un equipo sin necesidad de utilizar un disco duro, implementándola en un lápiz USB. Esto nos permite trabajar con cualquier equipo, por muy antiguo que sea.

Todas las minidistribuciones poseen unas características comunes:

- El uso del disco duro no es estrictamente necesario
- Utilizarán la memoria RAM como sistema de ficheros: /dev/ram-n
- Mínima ocupación de recursos: Entre 4 y 8 Mb de RAM
- Podremos instalarlas desde GNU/Linux, MS-DOS o sin sistema operativo mediante los sitemas LiveCD.
- Posibilidad de utilizer discos auxiliaries para aumentar sus funcionalidades.
- Mínima capacidad: entre 1Mb y 50Mb
- Procesador i386

Las minidistribuciones nacen con la idea de resolver una necesidad. En nuestro caso el principal objetivo es **salvar equipos antiguos sin disco y sin usar**. No todos los equipos soportan todas las distribuciones. Habrá que prestar especial atención en los requisitos de cada una de ellas y en función de lo que estemos buscando podremos elegir una u otra.

La tarea de elegir una distribución entre todas las que existen no es una tarea nada fácil. Es importante tener en cuenta lo que ofrece cada una de ellas. Las hay más pesadas y con más opciones. Más ligeras y con menos funcionalidades. Más visuales, sin entorno gráfico…

Tal y como hemos comentado la lista de distribuciones existentes en la red es bastante amplia. Podemos encontrar las mejores distribuciones de Linux para principiantes, para usuarios avanzados, con aspecto similar al de algún sistema operativo conocido, con alguna funcionalidad o como es nuestro caso, **las más ligeras para equipos antiguos:**

5.1.1 Distribuciones ligeras

Puppy Linux

Cuando hablamos de distribuciones ligeras, Puppy Linux no puede faltar. Se creó pensando en una distribución que ocupe los mínimos recursos posibles.

- Su imagen ocupa aproximádamente 100 MB en nuestro disco.
- Está disponible en version para LiveUSB o LiveCD.
- Cuenta con un repositorio donde podemos descargar los paquetes que necesitemos.
- Gracias a su interfaz gráfica es una distribución fácil de usar y no se necesita ser un experto en Linux para poder trabajar con ella.

En cuanto a los requisitos necesarios:

- Con 64MB de RAM y un procesador de 486 puede arrancar la distribución.
- Permite crear particiones SWAP.

Actualmente se encuentra disponible una version de Puppy Linux basada en Ubuntu y otra basada en Slackware. Elegir una u otra dependerá de la preferencia del Usuario.

Adjuntamos captura del estilo de su escritorio:

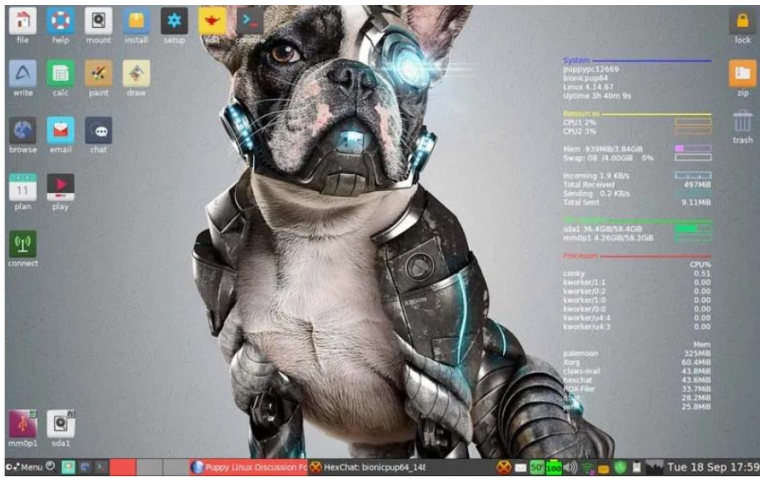

Figura 10: Escritorio Puppy Linux

Podemos encontrar toda la información necesaria sobre esta distro en su página official [10].

Lubuntu

El software que viene preinstalado en la distribución está pensado para consumir pocos recursos. Al estar basada en Ubuntu cuenta con una gran cantidad de documentación y foros donde encontrar información. Se encuentra en constante actualización. Cuenta con un entorno de escritorio LXDE.

Refiriéndonos a los recursos necesarios:

- Necesita mínimo 256 MB de RAM para usar los servicios básicos.
- Necesita una CPU Pentium II o Pentium III con unos 400Mhz de frecuencia.

En la siguiente captura se puede ver el estilo de su escritorio:

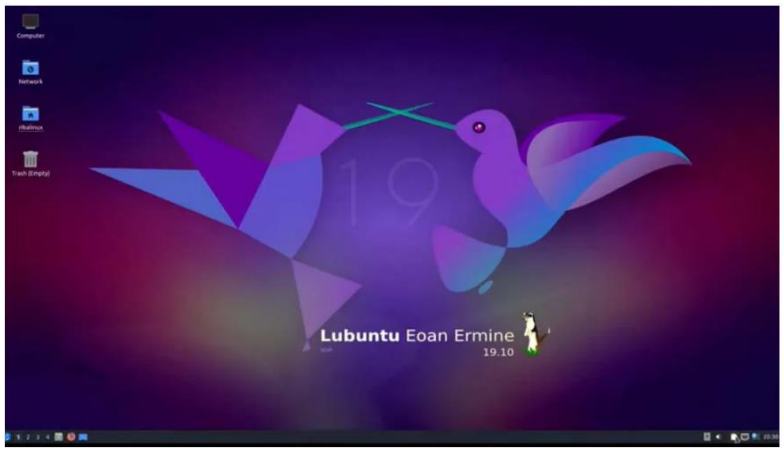

Figura 11: Escritorio en Lubuntu

Podemos encontrar toda la información necesaria sobre esta distro en su página official [11]

Linux Lite

Algunas de sus características principales son:

- Se trata de una distribución basada en Ubuntu LTS.
- Su interfaz gráfica facilita su uso y pueden trabajar con ella principiantes en Linux.
- Cuenta con un entorno de escritorio XFCE, ligero y su diseño puede recorder a Windows XP.

Para que podamos hacernos una idea de su diseño, adjuntamos captura de su escritorio:

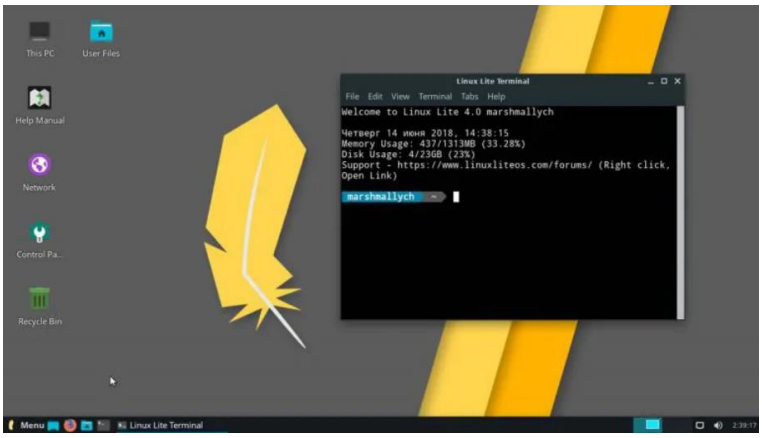

Figura 12: Escritorio en Linux Lite

Podemos encontrar toda la información necesaria sobre esta distro en [12]

Peppermint OS

Destacando algunas de sus características:

- Distribución basada en Lubuntu
- Entorno de escritorio LDXE
- Capacidades orientadas a la computación en la nube
- Permite acceder a los repositorios de Ubuntu

Su escritorio tendrá el siguiente diseño:

Figura 13: Escritorio en Peppermint OS

Podemos encontrar toda la información necesaria sobre esta distro en [13].

Información de todas las distribuidas obtenida de [14] y [15].

5.1.2 Distribución elegida: ALPINE LINUX

Cualquiera de las minidistribuciones explicadas en el apartado anterior hubiesen sido válidas para el proyecto. Sin embargo, después de probar muchas de ellas, la distribución finalmente elegida es Alpine Linux.

Figura 14: Logo distribución ALPINE LINUX

¿Y por qué Alpine Linux frente al resto? Según [16] :

- Es una distribución basada en **Linux, independiente, software libre, código abierto y de propósito general**. Fue diseñada para usuarios con experiencia en Linux, ya que al principio no es tan intuitiva como el resto de distros.
- Su página oficial la resume como **"SMALL. SIMPLE. SECURE"** o "PEQUEÑA. SIMPLE. SEGURA", refiriéndose a sus principales características ya que tiene como objetivo ser ligera y segura.
- Al estar construido alrededor de *musl libc y busybox* permite que consuma menos recursos que el resto de distribuciones de Linux. Su contenedor no requiere más de 8MB y para instalarse en algún disco necesita aproximadamente 130MB de almacenamiento, garantizando ser una distribución muy ligera.
- No obstante, cuenta con un repositorio muy amplio donde se recogen algunos paquetes de herramientas, aplicaciones y actualizaciones. Contiene los paquetes más utilizados como Firefox, GNOME, etc. Durante el desarrollo de nuestra distribución iremos agregando algunos de estos paquetes y al instalarlos, aumentará la capacidad de la distribución.
- Además, cuenta con su propio administrador de paquetes llamado apk y un sistema de inicio de OpenRC.
- Tal y como hemos comentado, Alpine Linux fue diseñado teniendo en cuenta la seguridad. Todos los archivos binarios se compilan como PIE (Position Independent Executables) con protección de destrucción de pila (stack-smashing) para mitigar los desbordamientos de búfer del usuario. **Estas características de seguridad proactivas evitan las vulnerabilidades.**

Una vez mencionadas las características generales de nuestra distribución, nos centraremos en por qué es la más adecuada para nuestro proyecto.

- Ocupa muy poco espacio en el disco.
- La distribución soporta ser instalada en un USB de forma persistente. Son muy pocas las que soportan esta opción, y para nosotros era un requerimiento imprescindible.

- No podemos olvidar que fue diseñada principalmente para routers x86, cortafuegos y **servidores** entre ellos. La idea de este proyecto es salvar un equipo antiguo con esta distribución de poca capacidad y que actue de servidor. Por lo que es un gran punto a favor.
- Cuenta con una gran wiki [17] donde es sencillo y rápido encontrar información acerca de instalación de paquetes, uso, etc. Punto muy importante para este estudio.
- En su repositorio se encuentra el paquete SQUID y es el proxy web que vamos a utilizar para nuestro proyecto, por lo que era indispensable que estuviese disponible.
- Recién instalada la distro de su página oficial, no cuenta con interfaz gráfica. Esto hace que ocupe aún menos espacio y además que podamos cumplir otro de los objetivos de este TFG: aprender a manejarnos con la consola de comando de Linux.

5.1.3 Requisitos del sistema

Los requisitos básicos para instalar Alpine Linux sin interfaz gráfica son:

- Al menos 128 MB de RAM
- Al menos 1 GB de almacenamiento de espacio en el USB donde vamos a instalarla.

5.2 Tecnologías utilizadas para crear la herramienta de control parental

A continuación explicaremos brevemente las diferentes tecnologías que hemos utilizado para crear nuestra herramienta de control parental.

5.2.1 Spring Tool Suite 4

Spring Tool Suite es un entorno de desarrollo basado en la versión **Java EE de Eclipse**. Gracias a la tecnología utilizada, **Spring Boot**, se fomenta la creación sencilla de aplicaciones Java.

Figura 15: Logo Spring Framework

Tiene como finalidad resolver, agilizar y manejar los problemas y complicaciones que van surgiendo durante la fase de programación. Gracias a sus módulos y extensiones personalizaremos la herramienta cumpliendo nuestras especificaciones.

¿Por qué Spring Framework frente a otras arquitecturas de desarrollo?

- Principalmente tenía que ser un Framework compatible con Frameworks web y nos ofrece **Spring MVC** para ello. Facilita este tipo de programación con la oportunidad de ir implementando todas las dependencias que vayamos necesitando.
- Además, queríamos crear una aplicación web segura. Gracias al módulo de Spring Security (muy utilizado en entornos empresariales por su seguridad) podemos cumplir esta especificación.
- Al ser tan utilizada por programadores Java, fue más intuitivo que en otros entornos encontrar la

solución a los errores que iban surgiendo, ya que a alguien le había pasado antes que a mi. Posee una comunidad muy activa que facilita documentación.

- Al tratarse de un diseño modular, nos permitía centrarnos en las clases que nos interesaban e ignorar el resto.
- **Posee plantillas** totalmente disponibles para utilizar en nuestro código, facilitándonos el trabajo.

Un concepto importante que no podemos confundir es que Spring Tool Suite es la herramienta de desarrollo. Sin embargo, **Spring Boot es la tecnología** que hemos utilizado para la generación de la aplicación web.

Para más información sobre Spring, recomendamos visitar su página web oficial [18].

Necesitábamos una aplicación web para que el usuario pudiese modificar el fichero de configuración del proxy (Squid en nuestro caso) de forma intuitiva y sin que necesitase conocimientos técnicos sobre este tema. Por ello, para la escritura del fichero de configuración de Squid, he desarrollado la aplicación web a través de Spring Boot. Las características a destacar de Spring Boot son:

- Nos permite **configurar automáticamente** Spring.
- Gracias a la tecnología **Maven** con su fichero **pom.xml** nos simplificará la configuración de la aplicación.
- Incorpora servidores como Apache o **Tomcat.**

Estas características se explicarán con más detalle en el Apartado 6. Pero para introducir un poco el proyecto, gracias a esta arquitectura de desarrollo, no hemos tenido que configurar como tal un servidor web. Se ha utilizado uno de los ofrecidos por Spring: Tomcat, adaptándolo a nuestra aplicación. En definitiva, **el usuario se comunica con la aplicación web gracias al servidor web Tomcat que provee Spring.**

Utilizando **HTTPS** como protocolo seguro, para asegurar la privacidad de las comunicaciones, el cliente, a través de un navegador web, accederá a la aplicación web en la URL https://[IP_servidor]/menu, donde la aplicación web ofrecerá un menú principal por el cuál desplazarnos por la misma.

A su vez, usando **Spring Security** (un módulo a destacar en Spring), se consigue que solo los usuarios correctamente autenticados puedan acceder a la aplicación, evitando así la manipulación malintencionada del fichero de configuración de Squid.

En el **Anexo B** se explicará con más detalle como instalar el entorno y en el apartado 5.2.5 la implementación de Spring en nuestro proyecto.

5.2.2 Docker

Docker es una tecnología de código abierto. Gracias a su tecnología empaquetaremos en contenedores independientes nuestra aplicación desarrollada en Spring (en la máquina virtual Linux) y automatizará su despliegue en Alpine Linux.

Figura 16: Logo Docker

Al ser Docker **independiente del sistema operativo**, no es necesario preocuparse de si el código funcionará en nuestra máquina, ya que si en nuestro entorno funciona, en nuestro equipo también lo hará. Docker permitirá que la aplicación funcione de forma autónoma.
En el Anexo B se explicará como se ha instalado Docker. Para entender el funcionamiento básico de Docker tenemos que tener en mente varios conceptos:

- En primer lugar, Docker es un **sistema de virtualización**. Esto a grandes rasgos no es más que simular un proyecto que en otra circunstancia ocuparía un hardware específico. Con Docker no necesitaremos ese hardware ya que se puede virtualizar y podremos visualizarlo de la misma manera.
- El concepto de **contenedor**. Cuando se utiliza Docker, su principal objetivo es crear contenedores ligeros y portables para que los proyectos puedan ejecutarse en cualquier máquina que tenga Docker instalado (como hemos dicho antes, independiente del sistema operativo). Esto simplifica el despliegue de la aplicación, ya que no habrá problemas de dependencias. Al fin y al cabo, un contenedor es "autocontenido", o con otras palabras: lleva todo lo que necesita para trabajar (versión de Java, Maven, servidor de aplicaciones…). Al usar Docker, únicamente nos preocuparemos de que funcione nuestro código, no de dónde se ejecutará.

Para más información sobre Docker, recomendamos visitar su página web oficial [19].

Si recordamos el escenario de nuestro proyecto, la finalidad es poder **desplegar la aplicación web en un equipo antiguo** con una **minidistribucion ligera sin interfaz gráfica**, Alpine Linux. Además, al querer que sea lo más ligera posible, no es una buena idea crear la aplicación web desde ahí ya que todo serían desventajas. Por eso, lo que se hará será crear toda la aplicación en otro equipo, con una máquina virtual Linux, y cuando se compruebe que todo funcione correctamente, utilizaremos **Docker para empaquetar la aplicación y desplegarla en Alpine Linux.** Cuando hablamos de "empaquetar" nos referimos a crear ese "contenedor" que contendrá la aplicación web y todos los recursos que necesita para funcionar.
Docker ha hecho que esta alternativa sea posible y nos ha facilitado el trabajo. Además, los contenedores de Docker son ligeros, por lo que no utilizan mucho espacio en la distribución.

Una vez entendidos los conceptos de "contenedor" y "sistema de virtualización", podemos profundizar en otros conceptos que también son importantes para saber cómo se ha ido desarrollando el proyecto: imágenes, Dockerfile y volúmenes:

Imágenes

Podríamos definir una imagen en Docker como **"captura del estado de un contenedor".** Una imagen puede contener desde un sistema operativo, hasta una pequeña aplicación. Utilizaremos estas imágenes para crear los diferentes contenedores.

En el propio registro de Docker podemos encontrar muchas imágenes base, de dónde podemos obtener todos los recursos que necesitemos.

Las imágenes se reconocen por estar compuestas por un ID y un nombre-versión, por ejemplo: Ubuntu:latest

Dockerfile

Para crear las imágenes mencionadas, necesitamos un fichero denominado **Dockerfile**. Será en este archivo donde escribamos todo lo que queremos que tenga la imagen. Estará compuesto por imágenes, comandos para instalar herramientas… Al final con este fichero, crearemos una nueva imagen modificando una imagen base.

En el **Apartado 6** explicaremos paso a paso el contenido del Dockerfile del proyecto.

Contenedor

Los contenedores, ampliando el concepto explicado previamente, no es más que una **instancia en ejecución de una imagen** en concreto. Con una imagen, podemos ejecutar más de un contenedor. Esto se utiliza para crear "copias de seguridad". Si hacemos alguna modificación en un contenedor y nos arrepentimos, podemos volver a una versión anterior utilizando otro de los contenedores que contienen la imagen sin cambios.

El utilizar varios contenedores para un proyecto también nos puede servir, además de "para controlar versiones", para repartir la carga de trabajo. Se pueden distribuir los accesos a la aplicación, y cada contenedor tendrá menos carga de peticiones. Justo ahora, se explicará el concepto de volumen y entenderemos cómo "unir" estos contenedores.

Aplicándolo a nuestro proyecto, como la aplicación web es ligera, no necesitaremos más de un contenedor.

Volumen

Tenemos que recordar que un contenedor de Docker es independiente de la máquina donde se ejecute, entonces no está conectado como tal a la máquina. Con otras palabras, si necesitamos utilizar algún recurso de la máquina, además de lo que se encuentra en nuestro contenedor, tendremos que crear un volumen.

Si queremos compartir datos, entre el contenedor y la máquina, utilizaremos un volumen. También sirven para compartir datos entre contenedores (ya que en una única máquina podemos instalar más de un contenedor).

Los expertos recomiendan no guardar datos persistentes dentro de un contenedor y utilizar también para esto los volúmenes, ya que se encuentran fuera del contenedor.

En resumen: un volumen podemos definirlo como un "punto en común" entre el contenedor y la máquina. Todo lo que se encuentre ahí podrá utilizarlo tanto la máquina como el contenedor de Docker.

¿Y esto dónde podemos encontrarlo en el proyecto?

Pues bien, si recordamos la finalidad de la aplicación web, comprobamos que se utiliza para que el usuario bloquee ciertas páginas web para todos los dispositivos que se conecten a su red. Para ello, una vez que tenemos guardadas todas las páginas en un fichero de texto, se encargará el proxy Squid de rechazar su acceso. El proxy Squid no se encuentra en el contenedor de Docker, si no en la máquina donde se despliegue la aplicación (o en este caso, el Docker). Por tanto, necesitamos una conexión entre la aplicación y Squid. Aquí será donde entre el concepto de volumen y resuelva el problema.

En el apartado 6 se explicará como hemos utilizado exactamente Docker en nuestro proyecto.

Muchos de estos conceptos están inspirados en los conceptos que explican en el blog para principiantes [20].

5.2.3 Squid

Squid es un servidor proxy para web con caché basado en software libre publicado bajo licencia GPL. Está orientado principalmente a HTTP, por lo que cubre nuestras necesidades:

Figura 17: Logo Squid

Algunas de las funcionalidades que nos ofrece este servicio son:

- Posibillidad de controlar el acceso a páginas web aplicando reglas (orden de sentencias en el fichero de configuración). Por tanto, es capaz de filtrar los servicios.
- Gracias a los logs que registra, somos capaces de ver qué peticiones han sido solicitadas y cuáles cursadas.
- Al actuar de cortafuegos, mejorará la seguridad de nuestra aplicación.

Otro de los motivos por lo que se ha elegido Squid es porque es software libre y es muy fácil encontrar información y ejemplos de su uso. Existen otros proxys [21] con la funcionalidad parecida a la de Squid, como son:

Privoxy

Privoxy se trata de otro proxy web. Entre muchas de sus herramientas, nos ofrece: opciones de filtrado de URLs (de manera que podamos controlar la navegación por internet) o eliminar anuncios.

Varnish Cache

También de código abierto como Squid. Se trata de un acelerador de aplicaciones web o como muchos lo conocen, caché de proxy HTTP inversa. Se instala en el servidor web que utilicemos y se encarga de almacenar en caché copias de los recursos que se solicitan a través del servidor. De esta manera, podemos mejorar el rendimiento de aplicaciones web.

Polipo

Proxy web principalmente utilizado para filtrar. Gracias a su uso podemos mejorar la privacidad de nuetra red. Ocupa poca capacidad y se define como pequeño y rápido a la hora de almacenar en caché.

Proxomitron

Proxomitron es también un proxy web, gratuito y muy flexible. Mediante su fichero de configuración, podemos adaptarlo a nuestras necesidades. Es ligero, por lo que no ocupa mucho espacio.

Proximodo

Este proxy nació basándose en el proxy Proxomitron, por lo que se pueden usar los mismos filtros que en Proxomitron. Una gran desventaja que hace que muchos usuarios descarten esta opción, es que no es compatible con las conexiones SSL.

Pasamos a profundizar más en el **proxy Squid** con la información que aparece en el curso avanzado en [22].

Tenemos que tener claro que **Squid se encargará de escuchar todas las peticiones de los usuarios**, les enviará una **respuesta** (permitiendo o bloqueando el acceso) y además **guardará en su caché una copia**. Por eso, este tipo de proxy lo podemos englobar dentro del tipo **proxy-cache.**

Además de ser software libre, cuenta con varias formas de configuración para adaptarlo a nuestro proyecto. Cuenta con otros modos de funcionamiento: inverso, transparente, **filtro de contenido**…Es importante destacar que es rápido y eficaz. En nuestro caso, también actuará como filtro de contenido, ya que se encargará de bloquear el acceso a ciertas URLs.

El fichero de configuración se encuentra en **/etc/squid/squid.conf.** Por defecto, ya se encuentra configurado para funcionar. Además, cuenta con múltiples comentarios en muchas de las líneas explicando qué se hace en cada momento,

Los logs los encontraremos en **/var/log/squid.** Si hubiese algún error al iniciar el servicio o al pararlo, podemos encontrar los mensajes de error en /var/log/syslog o /var/log/messages.

Podemos controlar el acceso a nuestro servidor de diferentes formas: por IPs, MACs, horarios, login/password, listas de acceso… En este caso, no queremos controlar el acceso al servidor, ya que nuestra intención es que el servidor gestione todos los dispositivos que se encuentran en el mismo domicilio que él. Por tanto en nuestro

fichero de configuración mínimo tendrá que aparecer:

```
## Example rule allowing access from your local networks.
## Adapt to list your (internal) IP networks from where browsing
## should be allowed
acl localnet src 10.0.0.0/8  # RFC1918 possible internal network
acl localnet src 172.16.0.0/12      # RFC1918 possible internal network
acl localnet src 192.168.0.0/16     # RFC1918 possible internal network
## Allow anyone to use the proxy (you should lock this down to client
networks only!):
# acl localnet src all
## IPv6 local addresses:
acl localnet src fc00::/7       # RFC 4193 local private network range
acl localnet src fe80::/10       # RFC 4291 link-local (directly plugged)
       machines
```

Ahora solo nos faltaría configurar los puertos desde los que el usuario puede acceder a la página web. Por defecto, los valores que aparecen en el fichero de configuración son:

```
acl Safe_ports port 80              # http
acl Safe_ports port 21              # ftp
acl Safe_ports port 443             # https
acl Safe_ports port 70              # gopher
acl Safe_ports port 210             # waiss
acl Safe_ports port 1025-65535      # unregistered ports
acl Safe_ports port 280             # http-mgmt
acl Safe_ports port 488             # gss-http
acl Safe_ports port 591             # filemaker
acl Safe_ports port 777             # multiling http
```

Tenemos que tener en cuenta que hasta que no reiniciemos squid, no cargará nuestra nueva configuración. Para ello ejecutamos desde el terminal: **/etc/init.d/squid restart ó service squid restart**

Otro dato importante a tener en cuenta será el puerto en el que se encuentra escuchando nuestro proxy. En el proyecto se ha dejado el puerto estándar:

```
## Squid normally listens to port 3128
http_port 3128
```

Si añadimos al fichero de configuración la siguiente línea:

```
## Where does Squid log to?
#access_log /var/log/squid/access.log
```

Se guardarán todas las peticiones que se le hacen al proxy.

Una vez se ha configurado el acceso, pasamos a configurar **qué páginas web serán accesibles y cuáles no**. Para

ello, utilizaremos las sentencias **ACL**. Gracias a estas líneas, le indicaremos al proxy si decidimos aceptar o rechazar dichas páginas. Sin embargo, estas líneas por sí solas no consiguen bloquear o permitir el acceso, si no que necesitan ir acompañadas de las sentencias **http_access y http_deny**. Es muy importante el **orden** en el que se establecen las reglas. Tenemos que saber que las reglas se *leen de arriba abajo*. Es decir, si al principio escribimor un "allow todas-las-paginas", y un "deny Youtube" justo debajo, no se bloqueará el acceso a Youtube ya que primero se leerá el allow. Por esta razón, **primero bloquearemos los accesos específicos y luego permitiremos.**

Resumiendo, gracias a su tecnología, cuando el usuario solicite cualquier página, nos permitirá eliminar o modificar campos de la cabecera de peticiones HTTP y permitirá o restringirá su acceso.

Para más información sobre Squid, recomendamos visitar su página web oficial [23].

En el **Anexo B** se explicará con más detalle cómo instalarlo y en el **Apartado 6** encontraremos la configuración de Squid adaptada a nuestro sistema.

6 DESARROLLO DEL PROYECTO

Para entender cómo se ha desarrollado el proyecto, presentamos primero el escenario de forma resumida:

6.1 Escenario final

- Se han utilizado dos equipos:
    - **Equipo antiguo:** Equipo con más de 10 años. Gracias al arranque mediante el USB, contará con la mini-distribución Alpine Linux.
    - **Portatil:** Contará con el sistema operativo Windows 10 de 64bits y una máquina virtual con Ubuntu 18.04.3 LTS (Linux).
- **La aplicación web se ha desarrollado en Spring**. Spring se encuentra instalado en la máquina virtual. En el **Anexo B** podemos encontrar detalladamente los pasos para instalar la herramienta.
- Para probar que lo implementado es correcto, ejecutamos la aplicación en la propia máquina virtual y probamos a entrar desde Windows. Como la **función de la aplicación web es conseguir bloquear ciertas páginas web** de nuestra red, necesitaremos tener **instalado y configurado el proxy Squid** en la máquina virtual.
- Una vez funcione correctamente, utilizando **Docker**, **empaquetaremos la aplicación** para posteriormente desplegarla en Alpine Linux. Para trasladarla al otro equipo, cargaremos el Docker creado en su repositorio (Docker Hub) para que desde el otro dispositivo pueda descargarse mediante terminal.
- Por tanto, nos encontramos con Docker instalado tanto en el equipo con la **máquina virtual Linux** y en el equipo antiguo con **Alpine Linux**.
- En Alpine Linux, al igual que hicimos en la máquina virtual, activaremos y configuraremos el proxy Squid.
- Descargamos la versión de Docker del repositorio y la desplegamos en Alpine Linux.
- Una vez en este punto, podemos probar tanto desde Windows como desde Linux que la aplicación web funciona correctamente.

En la siguiente figura podemos encontrar los pasos resumidamente:

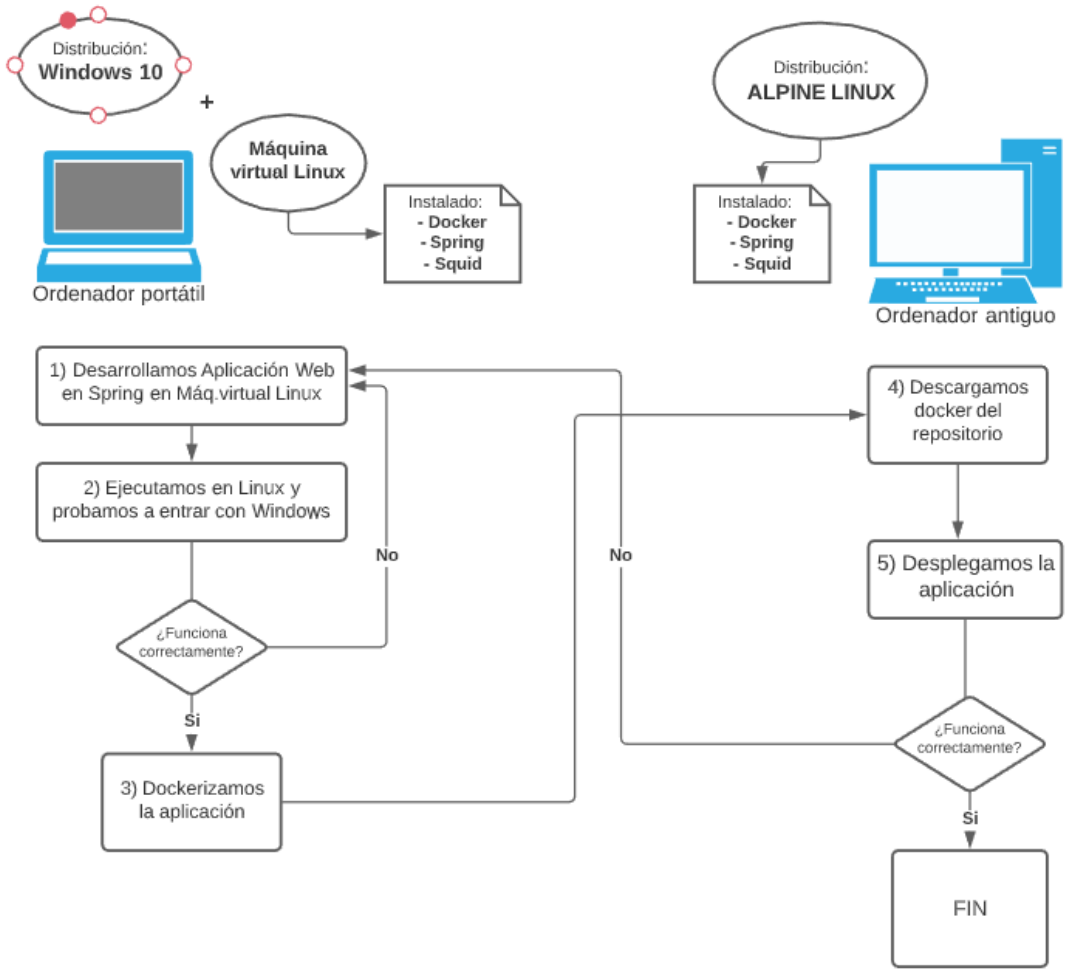

Figura 18: Escenario principal

6.2 Aplicación web con Spring Tool Suite

Se ha desarrollado una aplicación web a través de Spring Tool Suite utilizando ciertos módulos de Spring. Spring cuenta con numerosas tecnologías que nos facilitan la creación del proyecto. En la siguiente figura se aprecian algunas de las más comunes:

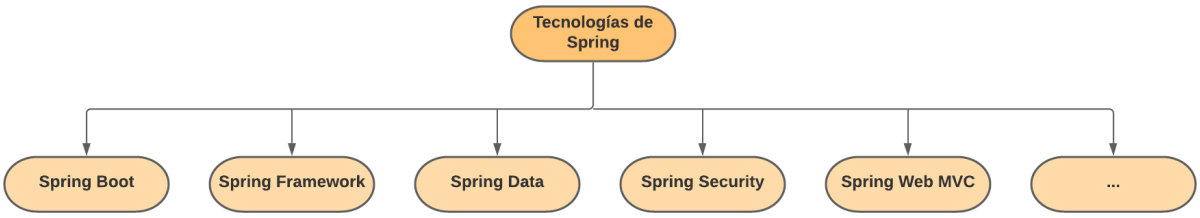

Figura 19: Tecnologías de Spring

Pasamos a explicarlas:

- **Spring Boot:** Sirve, entre otras cosas, para la autoconfiguración de Spring y la configuración del servidor web (Tomcat).

- **Spring Framework:** Nos provee de la infraestructura necesaria para el desarrollo de aplicaciones.
- **Spring Data:** Abstrae al programador del manejo de BBDD a través de distintas clases Java.
- **Spring Security:** Provee de autenticación y control de acceso a la aplicación desarrollada.
- **Spring Web MVC:** Aporta la infraestructura del patrón Modelo-Vista-Controlador a las aplicaciones web desarrolladas.

En el apartado 7 se explicará con más detalles las tecnologías utilizadas en este proyecto (Spring Boot, Spring Framework, Spring Security, Spring Web MVC).

La aplicación consiste, a grandes rasgos, en la gestión del proxy web Squid a través de una interfaz web sencilla. Permite únicamente a usuarios autenticados manejar el fichero de configuración del proxy añadiendo o eliminando nombres de páginas web. Todo esto se verá reflejado con diferentes opciones que irá ofreciendo la aplicación web en cada uno de sus menús.

6.3 Empaquetado del proyecto con Docker

Tal y como se explicó en el escenario final del proyecto, necesitamos tener instalado Docker en dos máquinas:

- En primer lugar, instalamos Docker en nuestra máquina virtual Linux, ya que es donde se ha desarrollado principalmente la aplicación Web. Una vez instalado, dockerizamos (empaquetamos) el proyecto y lo cargaremos en el repositorio official de Docker, Docker Hub.
- Después instalamos Docker en nuestro equipo antiguo con la distribución Alpine Linux. En este punto ya solo nos faltaría descargarnos la imagen del repositorio (Docker Hub) y desplegarla en la distro.

Una vez instalados (información en el **Anexo B**), pasamos a crear una nueva cuenta en el repositorio de Docker. Sera ahí donde vayamos subiendo todos nuestros proyectos. Es muy importante recordar el nombre de usuario, ya que sin él no podremos subir los ficheros. En la siguiente figura se observa la página Docker Hub:

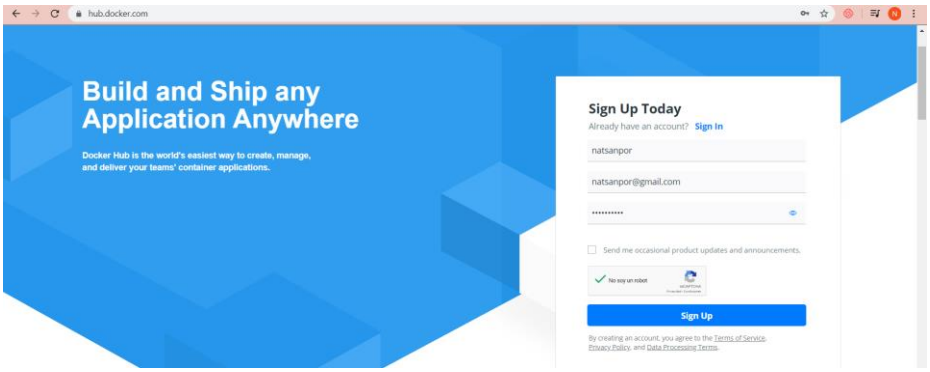

Figura 20: Pantalla inicial DockerHub

Como se ha mencionado, crearemos primero el Docker en Linux. Antes de comenzar con el proceso de empaquetado de la aplicación web, es importante comprobar que el servicio Docker está activo. Para ello ejecutamos el siguiente comando y nos fijamos en su salida: **"sudo systemctl status Docker"**

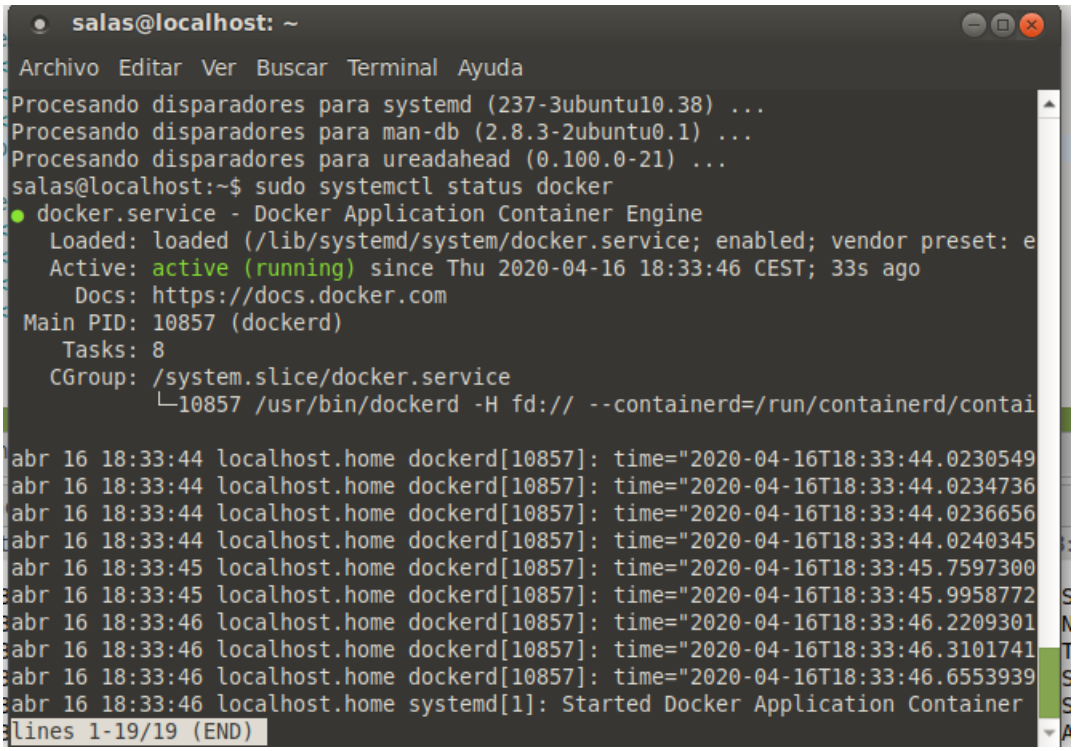

Figura 21: Comprobar estado de Docker en Linux

Efectivamente, su estado se encuentra en “active (running)”, por lo que pasamos a dockerizar la aplicación.

Como mencionamos cuando explicamos Spring aplicado al proyecto, vimos que la tecnología a utilizar es Maven. Por tanto, en primer lugar, en Spring ejecutamos lo siguiente:

1) Botón derecho sobre la carpeta principal del proyecto > **Run As > Maven Clean :** Así limpiamos todas las clases compiladas del proyecto y no tener problemas.
2) Botón derecho sobre la carpeta principal del proyecto > **Run As > Maven Install :** Instalamos el artefacto en el repositorio local.

Ahora sí la aplicación desarrollada en Spring se encuentra en condiciones de ser empaquetada. Para ello, el primer paso será crear nuestro fichero Dockerfile:

3) Botón derecho sobre la carpeta principal del proyecto > New > File . Creamos nuestro fichero Dockerfile:

(1) **FROM openjdk:8-jdk-alpine**
(2) **RUN addgroup -S spring && adduser -S spring -G spring**
(3) **RUN apk update**
(4) **RUN apk add --update ttf-dejavu && rm -rf /var/cache/apk/***
(5) **RUN apk add dbus**
(6) **RUN apk add cgmanager**
(7) **RUN apk add glib-networking**

(8) **RUN apk add fontconfig**
(9) **RUN apk add --no-cache nss**
(10) **RUN rm -rf /var/lib/apt/lists/***
(11) **USER spring:spring**
(12) **ADD target/serving-web-content-0.0.1-SNAPSHOT.jar user.jar**
(13) **ARG JAR_FILE=target/*.jar**
(14) **COPY ${JAR_FILE} app.jar**
(15) **ENTRYPOINT ["java","-jar","/app.jar"]**
(16) **ARG DEPENDENCY=src/main/resources**
(17) **COPY ${DEPENDENCY}/static/css .**
(18) **COPY ${DEPENDENCY}/static/jpg .**
(19) **COPY ${DEPENDENCY}/keystore .**
(20) **COPY ${DEPENDENCY}/application.properties .**

Significado de cada línea:

- (1): **FROM** indica que sistema operativo va a utilizar el contenedor. En este caso Alpine Linux.
- (2): **ADDGROUP** y **ADDUSER**: Creamos un usuario y grupo denominado Spring que utilizaremos posteriormente:
- (3): **RUN** ejecuta el siguiente comando y actualiza el índice de paquetes disponibles dentro de Alpine Linux.
- (4): Con esta línea también **eliminamos la "caché**", para no detectar recursos antiguos que no sirven.
- (5): Instala **"dbus"**, que es un sistema de comunicación de procesos, para poder comunicarse la aplicación con el servidor.
- (6): Instala "**cgmanager**", que no es más que un Daemon para gestionar programas que utilizan dbus.
- (7): Al instalar "**glib-networking**", podemos utilizar los módulos relacionados con la red.
- (8): Con "**fontconfiguration**", nos pemitirá utilizar el css correctamente en la aplicación web, ya que es una biblioteca diseñada para ofrecer una lista de fuentes.
- (9): Como queremos que el Docker ocupe lo mínimo posible, con la opción **–no-cache** permite no almacenar el caché el índice localmente.
- (10): Para seguir eliminando información innecesaria, **eliminamos la información del estado de cada recurso antiguo.**
- (11) **USER** indica el usuario y grupo que queremos que sea propietario y que ejecute el resto de comandos que quedan en el Dockerfile.
- (12), (13), (14), (15): Al final nuestra intención es crear una **aplicación JAR** para poder ejecutarla desde el terminar con algunos argumentos. Recordamos que los archivos JAR se utilizan para ejecutar

aplicaciones sin tener en cuenta en qué entorno de trabajo de java nos encontremos. Por tanto, cuando se genere la imagen Docker correspondiente, se copiará el JAR de nuestro directorio target a la imagen. Cuando arranquemos en Alpine Linux el contenedor, ejecutaremos el comando "java" que inicia nuestra aplicación web.

- (16): Para facilitar la estructura de las últimas líneas, creamos una variable con el nombre de la carpeta para luego hacer referencia a ella en sus subdirectorios.
- (17), (18), (19), (20): Nuestro contenedor necesitará recursos que se encuentran en esas carpetas, por lo que las importamos.

Una vez comprendido el contenido del Dockerfile, pasamos a crear el Docker:

4) En el terminal, dentro de la carpeta donde se encuentre el proyecto ejecutamos:

>> sudo docker build -t usuario/nombreDocker:mytag .

5) Si no ha ocurrido ningún error, ejecutamos posteriormente:

>> sudo docker push usuario/nombreDocker:mytag

De esta manera, se creará el Docker correspondiente a la aplicación. Con el paso nº 5, subiremos al repositorio de Docker (Docker hub) nuestro Docker, para que posteriormente podamos descargarlo desde el equipo antiguo con la distribución Alpine Linux.

6) En el terminal del equipo con Alpine Linux:

>> docker pull usuario/nombre:mytag

7) Para poner en marcha el servicio, desde Alpine Linux:

>> docker run --network="host" -v /root/CarpetaTFG/:/opt/CarpetaTFG -it usuario/nombreDocker:mytag bash -p 4443:4443

De esta manera, crearemos el volumen compartido entre Docker y la distribución para que pueda funcionar el servicio correctamente.

6.4 Configuración de Squid aplicada al proyecto

El principal fichero de configuración es /etc/squid/squid.conf, tal y como se explicó en el **Apartado 5.2.3 Squid**. En este apartado se indicarán qué cambios explícitos se han añadido en el fichero de configuración.

Las IPs desde las que se puede acceder a la página web se han dejado tal y como se indicó en el Apartado 5.2.3, ya que así, todos los dispositivos que se encuentren en la misma subred que el proxy podrán acceder.

El puerto en el que se encontrará escuchando el servidor que gestiona la aplicación web será el 4443. Sin embargo, como una de sus líneas configuradas es:

```
acl Safe_ports port 1025-65535      # unregistered ports
```

No será necesario añadir una línea adicional indicando explícitamente que el puerto es el 4443, ya que se encuentra incluido en ese rango.

El siguiente paso es configurar el orden de las sentencias acl para que nos bloquee y permita el acceso a ciertas páginas web. Para que el proxy web bloquee cualquier URL, necesitamos añadir las siguientes sentencias justo antes del comentario #And finally deny all… :

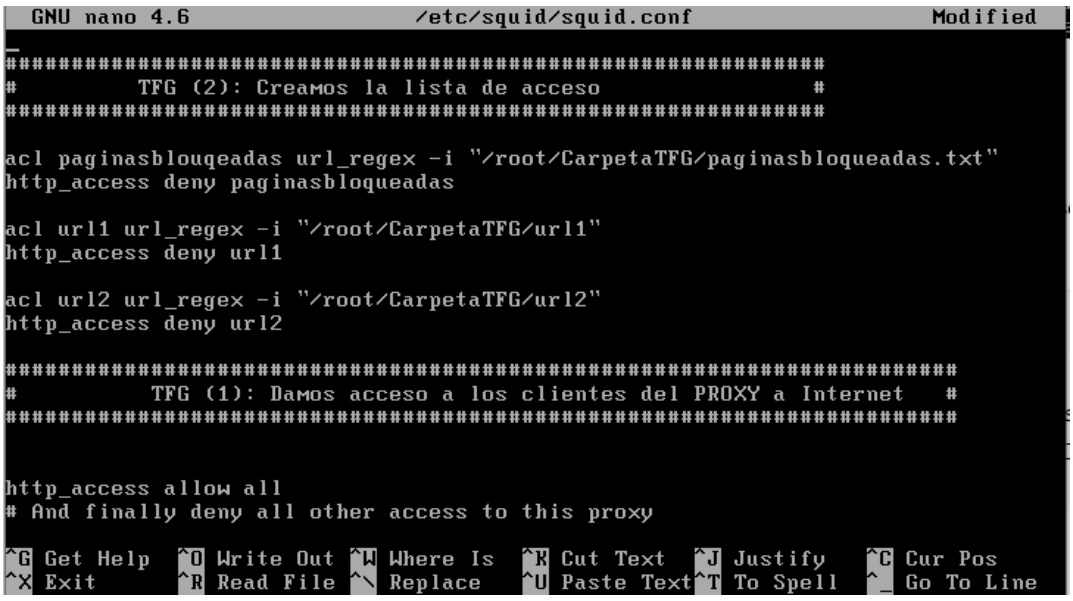

Figura 22: Modificación fichero de configuración squid.conf – Squid

Como se indicó en el Apartado 5.2.3, este tipo de sentencias se leen de arriba hacia abajo, por lo que primero tendremos que denegar las páginas específicas y luego permitir el resto. Como se indicó en el Apartado 6.1, todas las URLs que queramos bloquear se irán añadiendo en el fichero **paginasbloqueadas.txt**, por lo que habrá que añadirlo en el fichero de configuración junto con un "http_access deny" (tal y como se ve en la Figura 49).

Además, para mejorar el servicio que ofrece la aplicación web, ya vienen **explícitamente bloqueadas más de 10.000 URLs** procedentes de un fichero creado por especialistas que recomiendan bloquear ciertas páginas web. Su contenido se encuentra en los ficheros **url1 y url2,** y también se encuentran bloqueados (http_access deny). Se ha dividido en dos ficheros, ya que el procedimiento para añadir dichos ficheros txt en Alpine Linux (distribución final donde se usará el servicio) ha sido el siguiente:

- En la máquina Linux se descargó dicho fichero desde [33]
- El siguiente paso fue transmitir dicho fichero a la distribución Alpine Linux. Como hemos mencionado en varias ocasiones, se trata de una distribución ligera sin interfaz gráfica. Tampoco acepta memorias USB como dispositivo externo, por lo que la mejor forma de pasar el fichero fue utilizando la URL [34]. Esta página ofrece una nube donde podemos subir ciertos archivos y mediante una URL que

también proporciona nos permite descargar todos los ficheros que queramos en el dispositivp que sea.

- Al ser un fichero que ocupa mucho espacio, tuvimos que dividirlo en dos partes: url1 y url2.
- Para descargarlo en Alpine Linux, utilizamos el comando wget (antes se tuvo que descargar con apk add wget) y escribimos desde el terminal: wget link1-Obtenido-con-la-pagina y wget link2-Obtenido-con-la-pagina
- Movimos los ficheros descargados a la página dedicada para el TFG:
    - mv url1 /root/CarpetaTFG/url1
    - mv url2 /root/CarpetaTFG/url2
- Una vez en este punto, ya los ficheros se encuentran listos para ser bloqueados en el fichero de configuración de Squid (tal y como aparece en la Figura 49)
-

Para aplicar los cambios, siempre tenemos que reiniciar SQUID:

Figura 23: Reiniciar Squid

Si en el fichero "paginasbloqueadas.txt" añadimos Facebook, comprobamos que efectivamente nos bloquea el acceso:

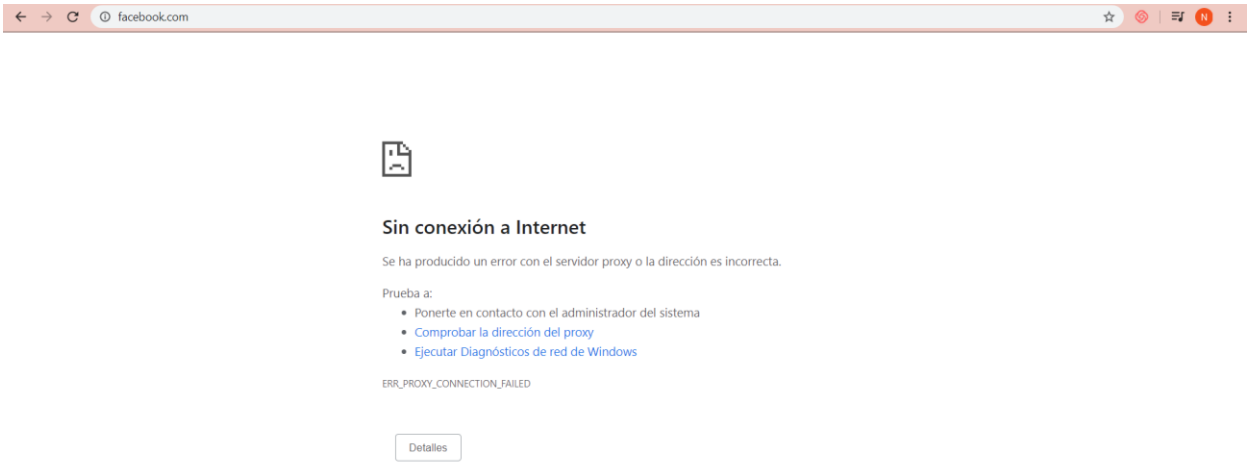

Figura 24: Ejemplo bloqueo URL - Squid

Tal y como se ha explicado previamente, cada vez que hay una actualización en la lista de páginas bloqueadas es necesario un reinicio del proxy. Para simplificar la tarea y no sobrecargar al sistema, hemos utilizado **crontab**. Crontab tiene una serie de scripts a ejecutar cada cierto tiempo.

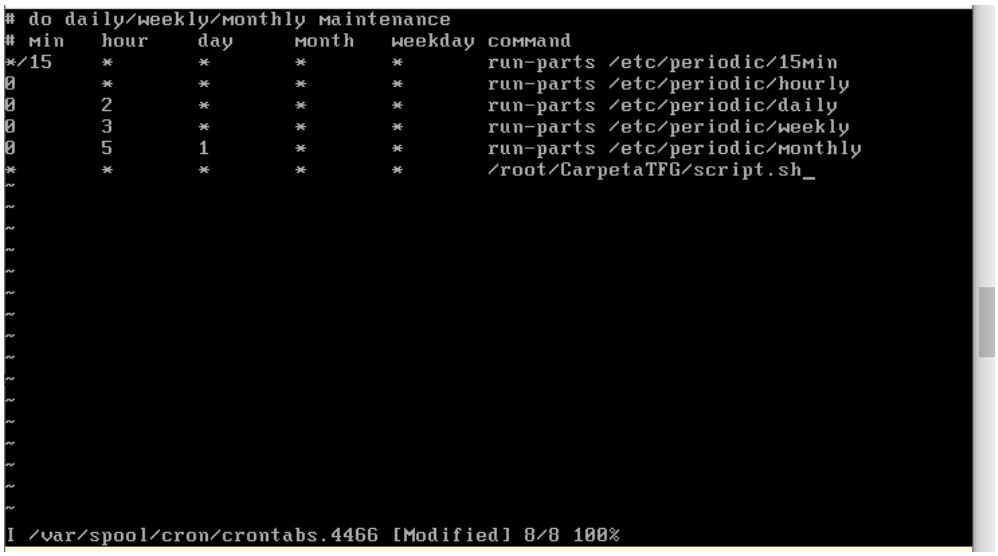

Figura 25: Contenido de crontab

Al ser ***** significa que el script.sh se ejecutará cada minuto. El contenido del script es:

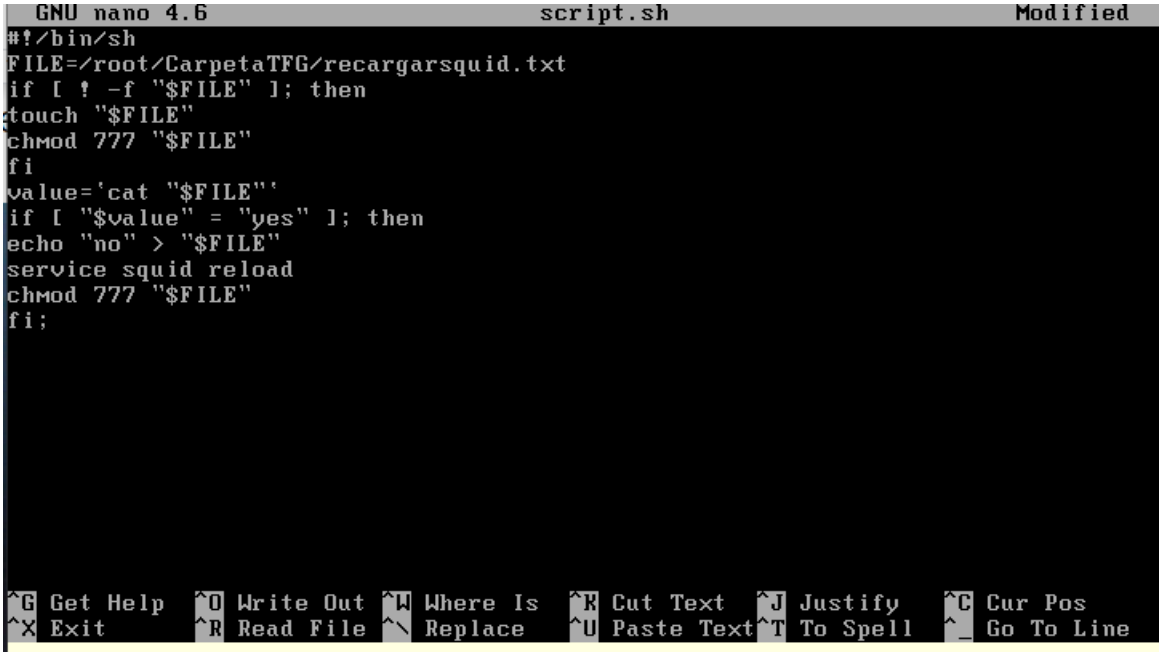

Figura 26: Script para reiniciar squid – script.sh

Dentro del directorio CarpetaTFG, hay un fichero "recargarsquid.txt" que actuará como bandera. Cada vez que un usuario escribe una página web a bloquear desde la aplicación, además de añadirse a la lista "paginasbloqueadas.txt", también escribe en el fichero "recargarsquid.txt" la palabra "yes". Entonces, este script se encargará de comprobar el contenido del fichero "recargarsquid.txt". Si se encuentra con "yes", reiniciará squid y sobreescribirá el valor a "no". Si por el contrario se encuentra directamente "no", no hará nada y esperará que en la próxima ejecución del script el valor haya cambiado.

De esta manera únicamente se recargará el proxy squid cuando sea estrictamente necesario.

7 APLICACIÓN WEB

Internet es mucho más que una tecnología. Es un medio de comunicación, de interacción y de organización social.

-Manuel Castells-

A la hora de crear nuestra aplicación web se hizo pensando en los usuarios finales que lo iban a utilizar. Además de que fuese atractivo y visual, queríamos que fuese simple. De un vistazo, el usuario pudiese encontrar lo que buscaba. También queríamos que fuese eficiente: cumpliera con lo esperado de una herramienta de control parental.

7.1 Interfaz web de la aplicación

7.1.1 Inicio

Para acceder a la aplicación web, escribimos en cualquier navegador la dirección IP y el puerto donde se encuentra escuchando nuestro servidor. En nuestro caso, las direcciones IP serán dinámicas ya que están configuradas mediante el protocolo DHCP. Por tanto, para acceder a la aplicación web: **https://IP_CLIENTE:4443**

Al cargar, aparecerá por pantalla la página de inicio:

Figura 27: Pantalla inicio superior – Aplicación Web

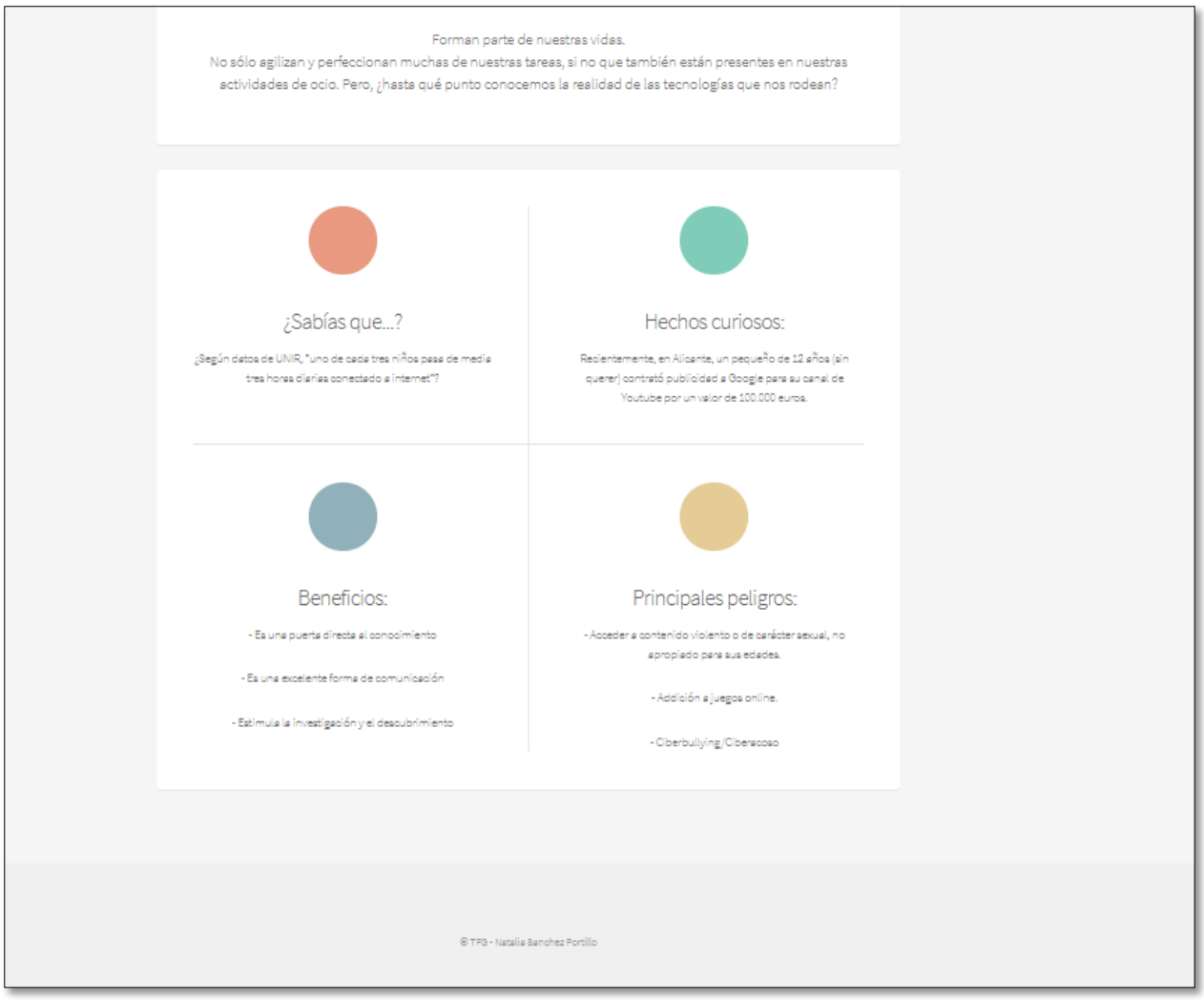

Figura 28: Pantalla inicio inferior – Aplicación Web

Podemos observar que la pantalla de inicio contiene una breve introducción de por qué es importante utilizar esta herramienta en nuestro hogar si viven menores. Menciona algunos hechos curiosos, beneficios y los principales peligros que puede acarrear Internet.

Antes de Iniciar Sesión en el servicio, nos da la posibilidad de pulsar en el botón “Contacto”

Contacto

Figura 29: Pantalla contacto superior – Aplicación Web

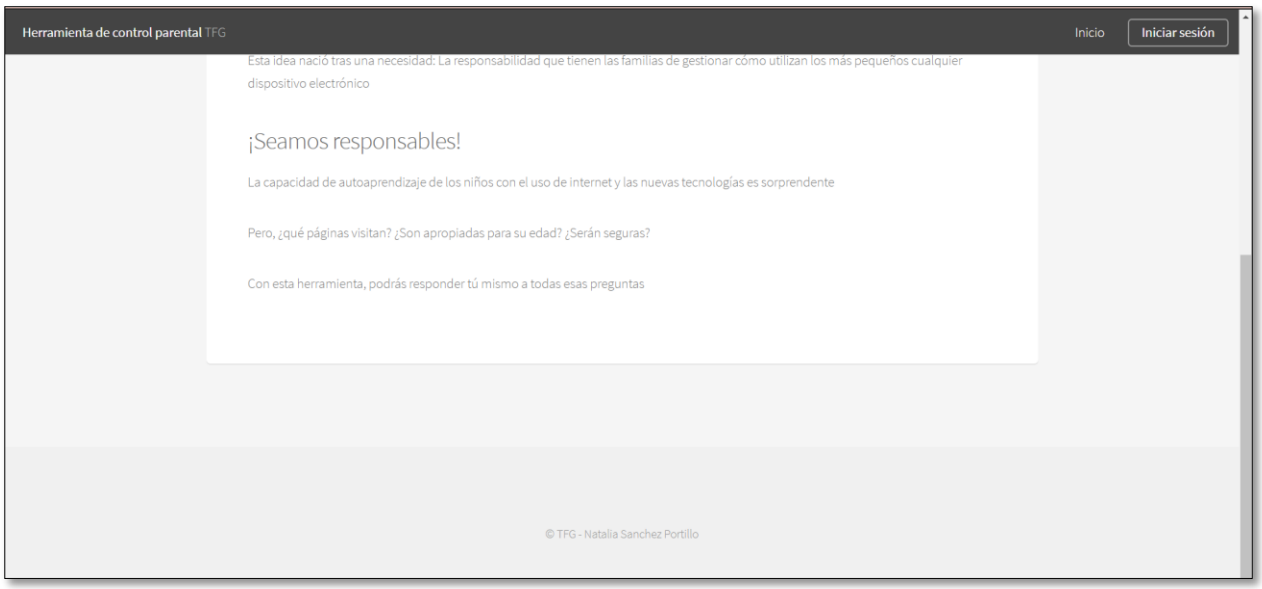

Figura 30: Pantalla contacto inferior – Aplicación Web

Podemos apreciar una breve introducción sobre por qué nace el proyecto y quiénes son los autores.

7.1.2 Iniciar Sesión

Para poder disfrutar de las opciones que ofrece el servicio, es necesario iniciar sesión:

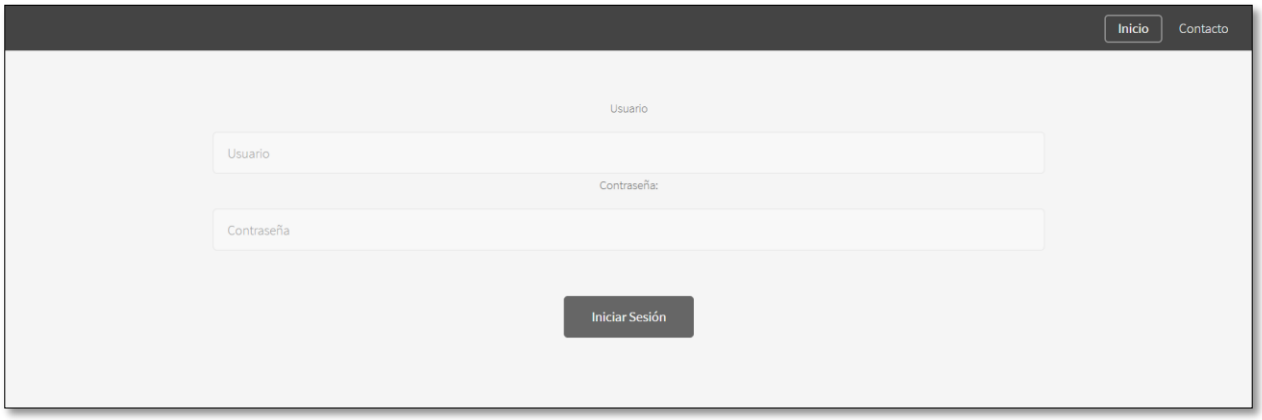

Figura 31: Pantalla inicio sesión – Aplicación Web

Comprobamos qué ocurre si el usuario y la contraseña son incorrectos:

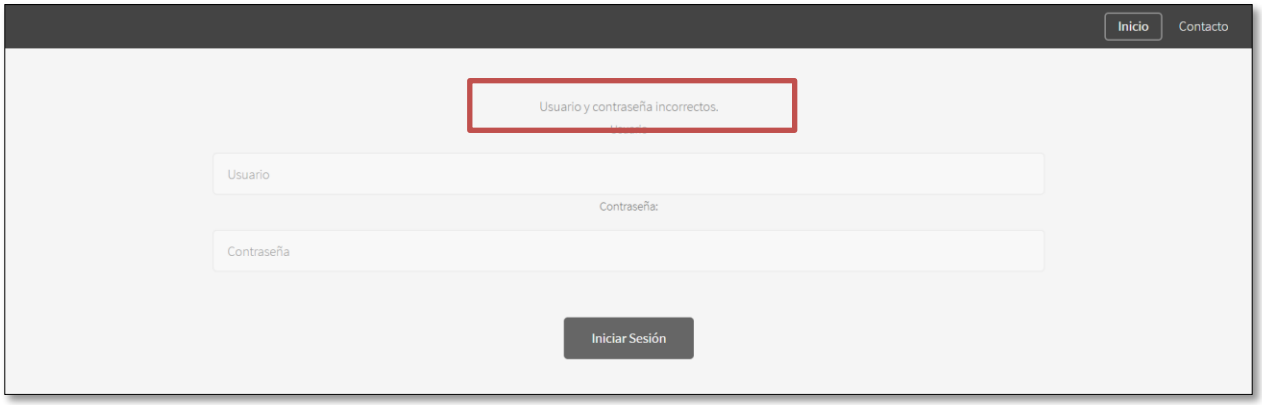

Figura 32: Pantalla mensaje de error inicio sesión – Aplicación Web

7.1.3 Menú principal

Una vez iniciado sesión correctamente, aparecerá por pantalla el menú principal. Como no puedo capturar la ventana entera en una única captura, paso a describir lo que contiene el menú:

- En la parte superior nos encontramos con los accesos directos a todas las opciones que ofrece la aplicación web:
    - Búsqueda
    - Bloquear
    - Desbloquear URLs
    - Lista de URLs
    - Bloquear URLs masivas

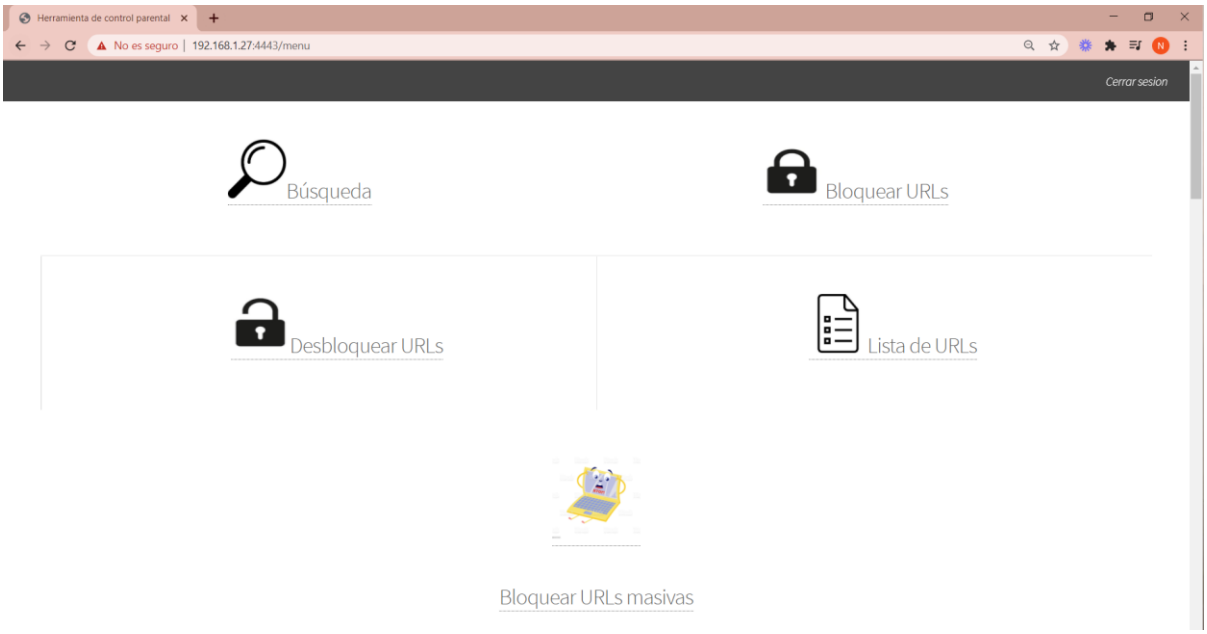

Figura 33: Menú principal (1) – Aplicación Web

- Si seguimos navegando en esta pantalla, al bajar, nos encontramos cada una de estas opciones con una pequeña descripción explicando lo que hacen:

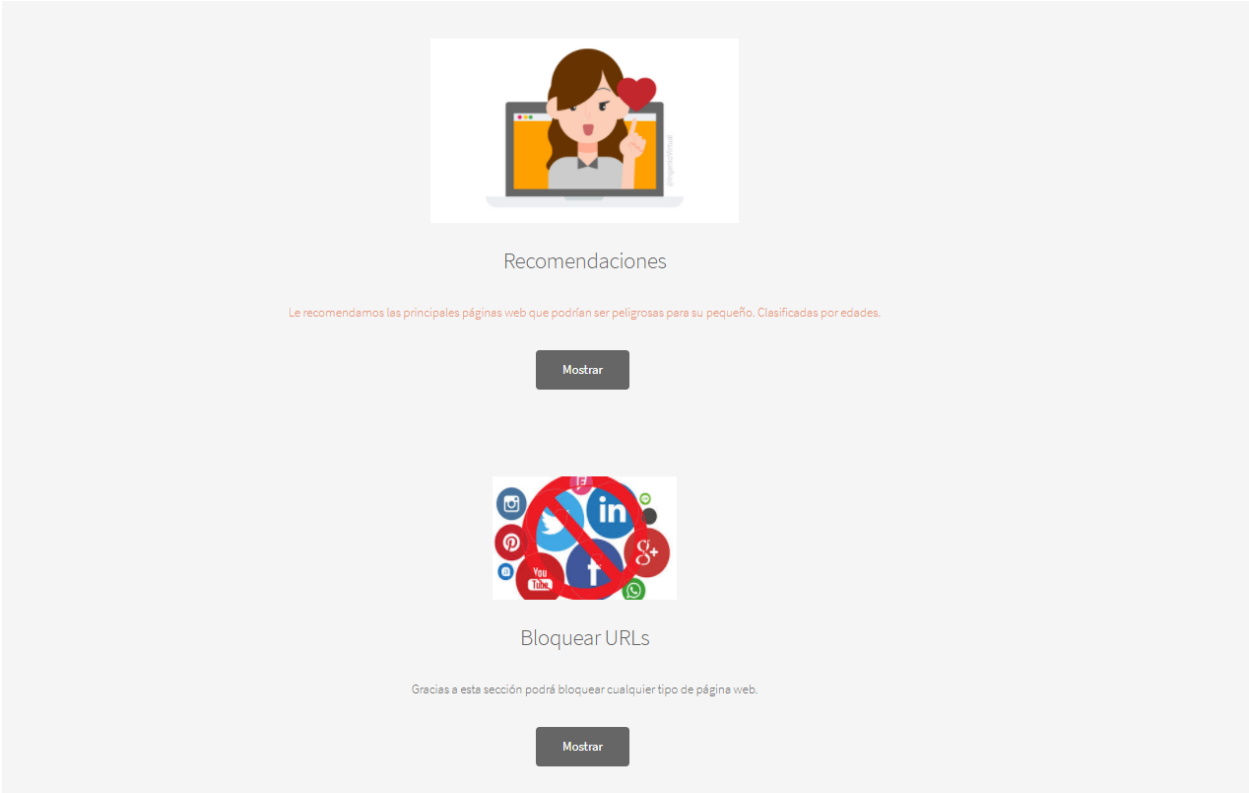

Figura 34: Menú principal (2) – Aplicación Web

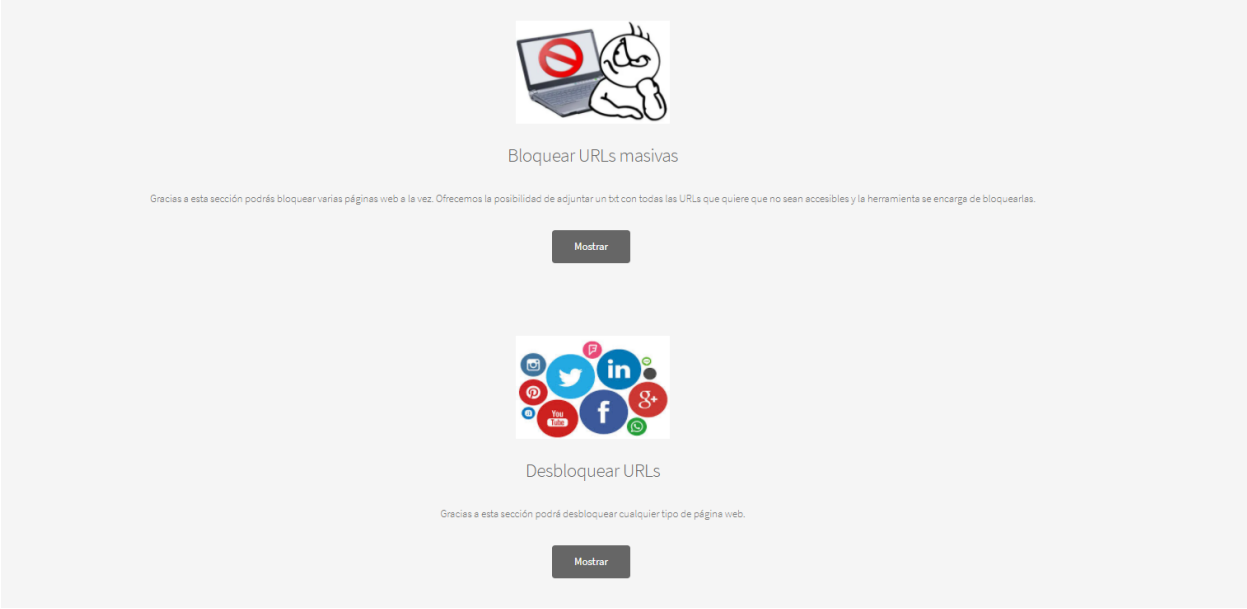

Figura 35: Menú principal (3) – Aplicación Web

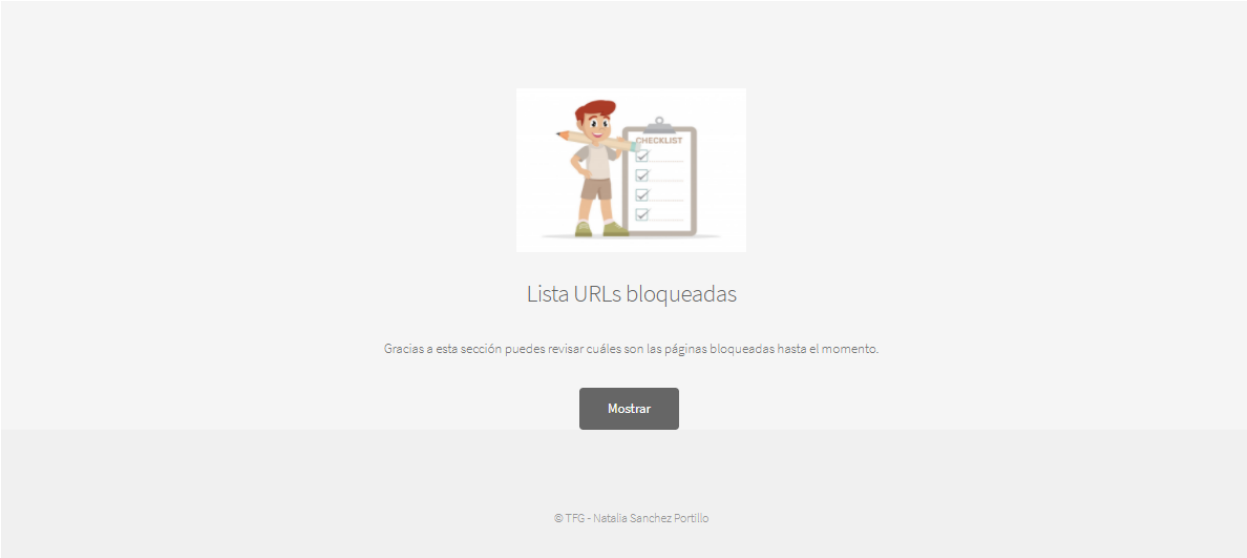

Figura 36: Menú principal (4) - Aplicación Web

Explicaremos una a una cada una de las opciones.

Recomendaciones

En esta opción podemos añadir nuevas páginas web a nuestra lista de URLs bloqueadas, por lo que los más pequeños no podrán acceder a ellas si lo intentasen. Es una manera fácil y directa de restringir el acceso sin tener que escribir, únicamente haciendo click sobre el icono que queramos.

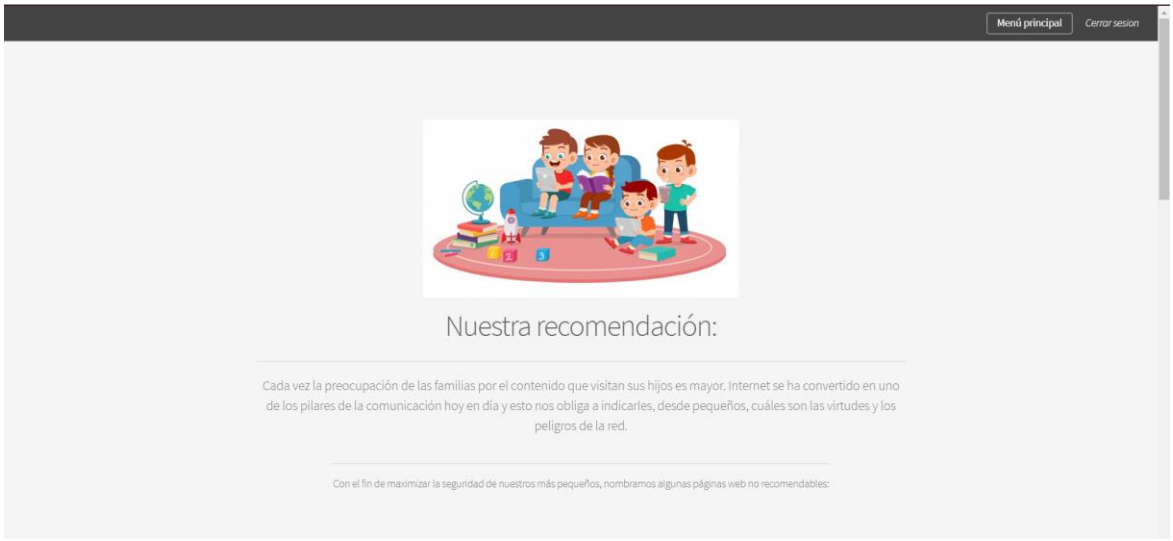

Figura 37: Pantalla recomendación (1) – Aplicación Web

Encontramos un apartado dedicado a las herramientas de citas y redes sociales:

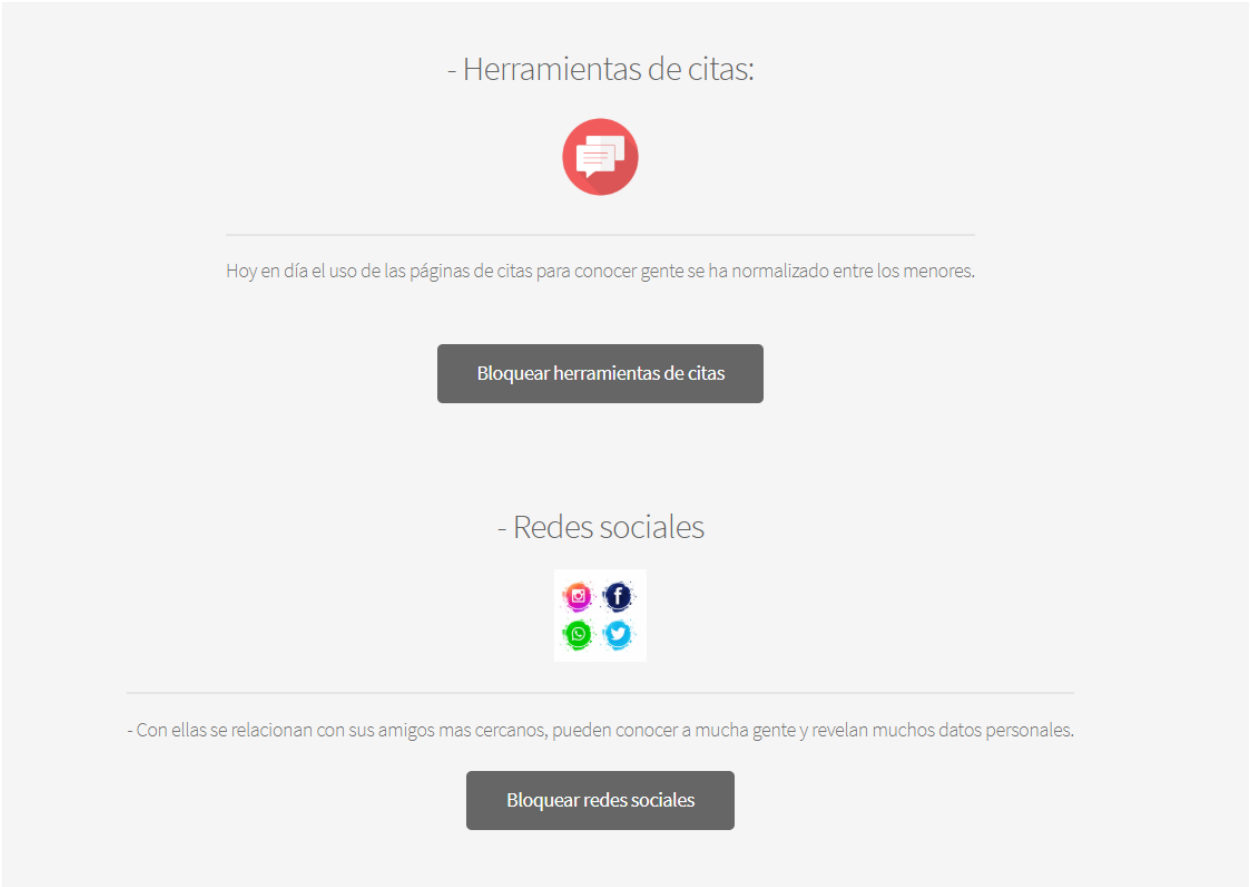

Figura 38: Pantalla recomendación (2) – Aplicación Web

La aplicación ya tiene integrado un fichero con más de 10.000 URLs que catalogan los expertos como inadecuadas o peligrosas. Por defecto, la aplicación ya bloquea todas esas páginas web.

Como podemos ver en la siguiente figura, también hay un apartado dedicado a las páginas de juegos:

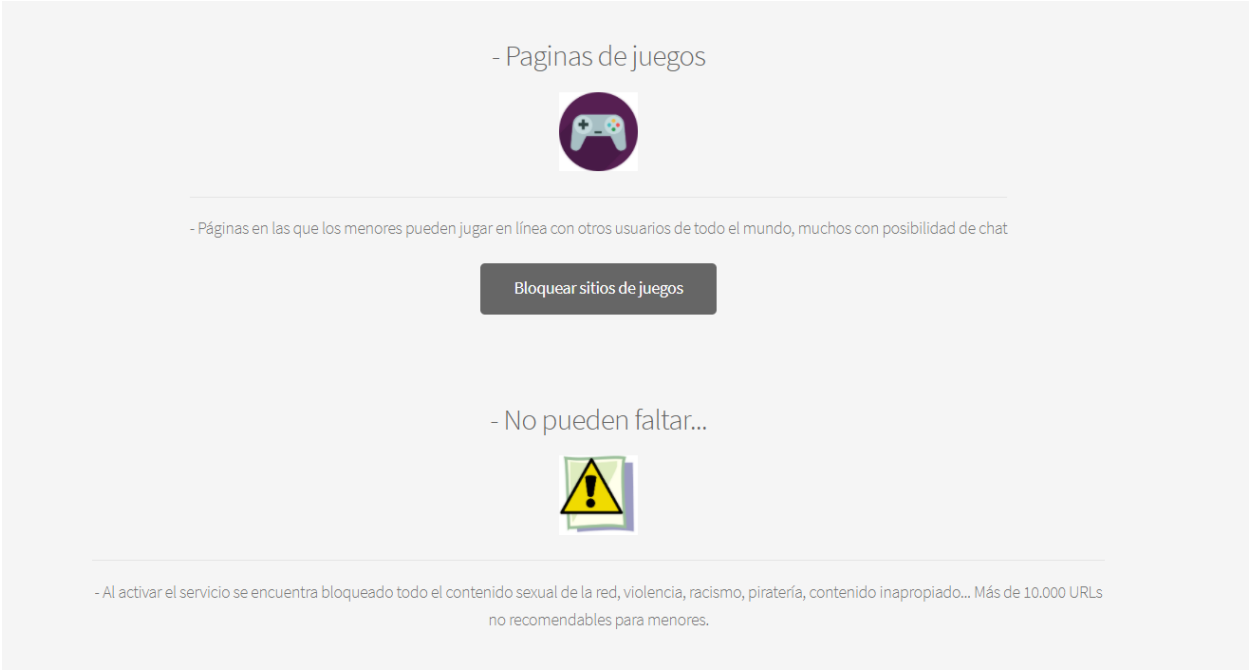

Figura 39: Pantalla recomendación (3) – Aplicación Web

Tanto el apartado de juegos, redes sociales y citas siguen el mismo formato.

Por ejemplo, si quisiésemos bloquear alguna red social, pulsaríamos sobre el botón "Bloquear redes sociales" y aparecería lo siguiente:

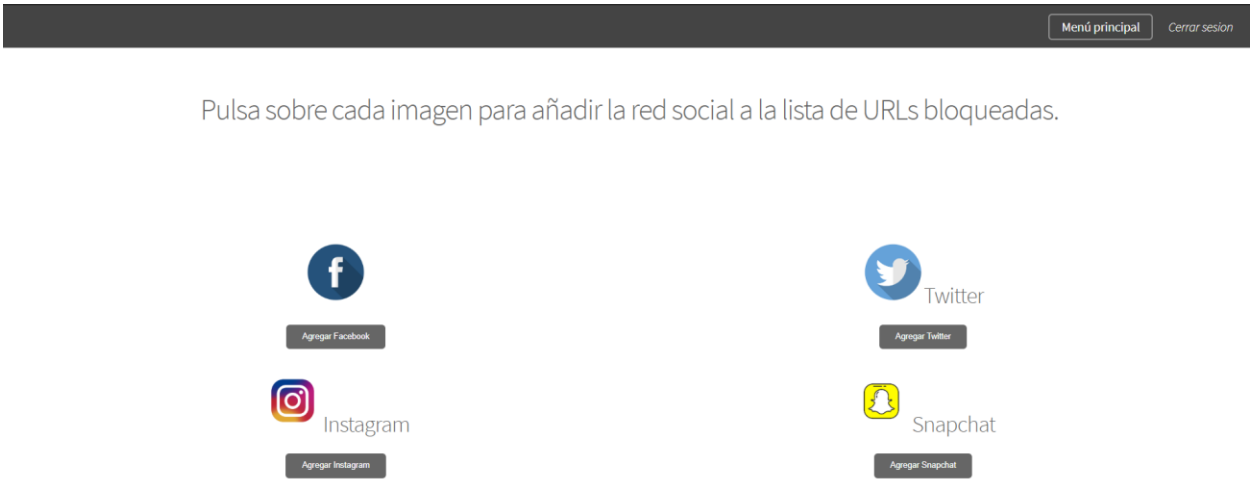

Figura 40: Pantalla recomendación Bloquear RRSS – Aplicación Web

Lo mismo pasa con las herramientas de citas:

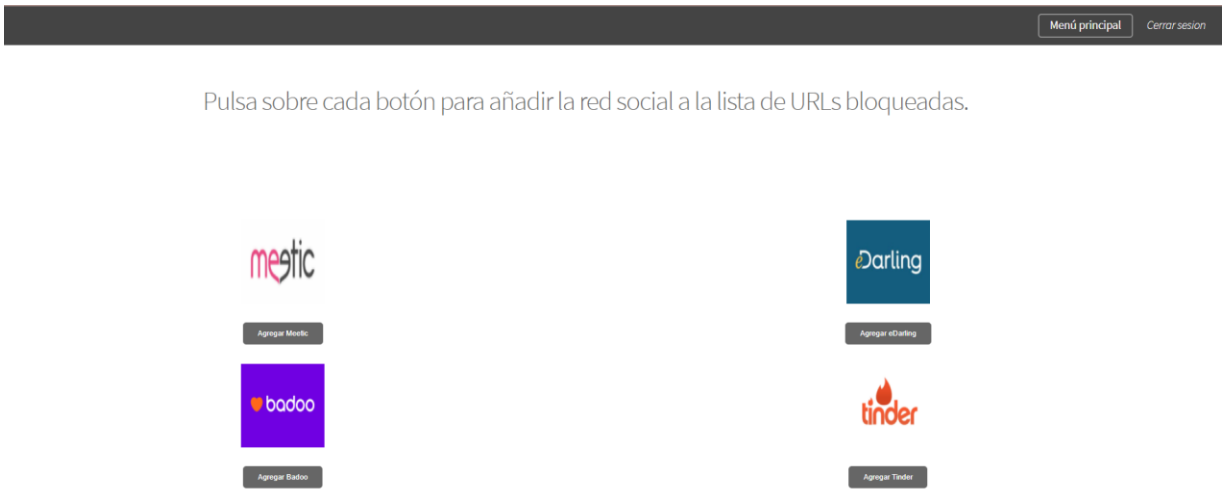

Figura 41: Pantalla recomendación Bloquear Citas - Aplicación Web

Y con las páginas de juegos:

Figura 42: Pantalla recomendación Bloquear Juegos - Aplicación Web

Bloquear páginas web

Para bloquear una página web, sólo tendremos que escribir su nombre en la barra que aparece:

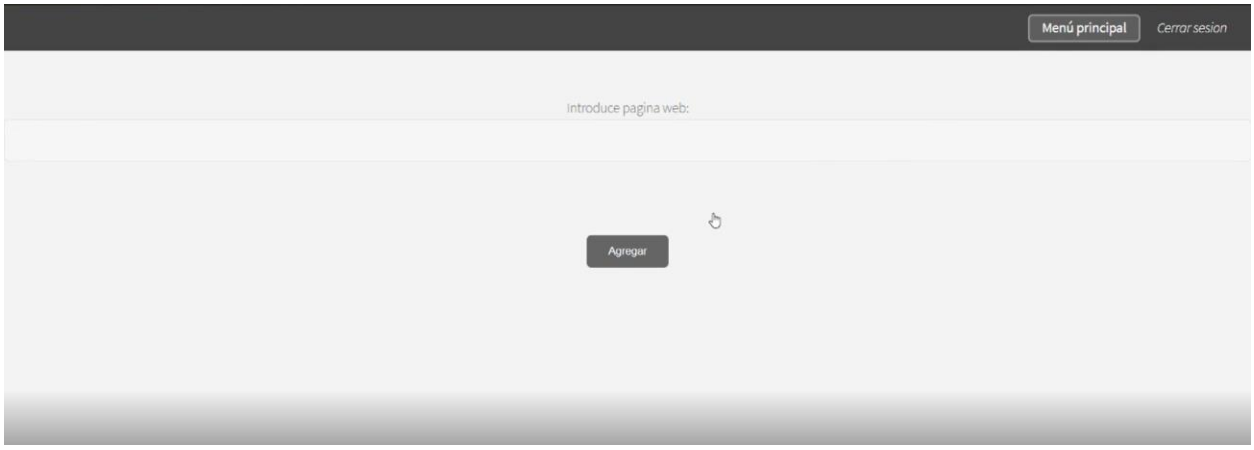

Figura 43: Pantalla bloquear páginas web – Aplicación Web

Desbloquear páginas web

El proceso de desbloqueo es muy parecido al de bloquear URLs. Para desbloquear una página web, únicamente tendremos que escribir su nombre en la barra que aparece:

Figura 44: Pantalla desbloquear páginas web – Aplicación Web

De esta manera, la página web se eliminará de la lista de URLs bloqueadas y se permitirá su acceso.

Bloquear URLs masivas

Si quisiésemos bloquear más de una URL a la vez, la aplicación nos ofrece la posibilidad de cargar un fichero txt con todas las páginas que no queremos permitir y ya se encarga de gestionarlas.

Figura 45: Pantalla bloquer URLs masivas - Aplicación Web

Lista páginas web bloqueadas

Gracias a esta opción podemos ver qué y cuántas páginas hemos bloqueado. Cualquier página que se encuentre en esta lista, será de acceso restringido.

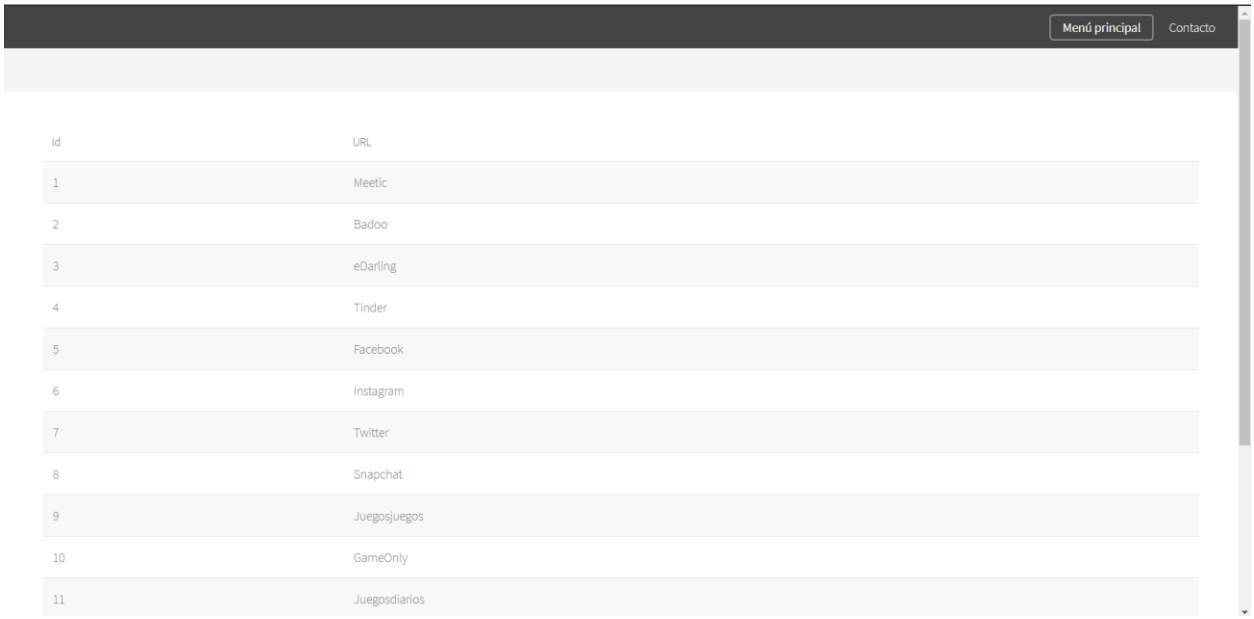

Figura 46: Lista de URLs bloqueadas – Aplicación Web

7.2 Desarrollo de la aplicación web en Spring

1) Como hemos explicado al iniciar este apartado, en primer lugar, desarrollaremos la aplicación web en la máquina virtual utilizando Spring. Una vez instalada la herramienta (**Anexo B**), para iniciarla simplemente ejecutamos desde el directorio donde se encuentre:

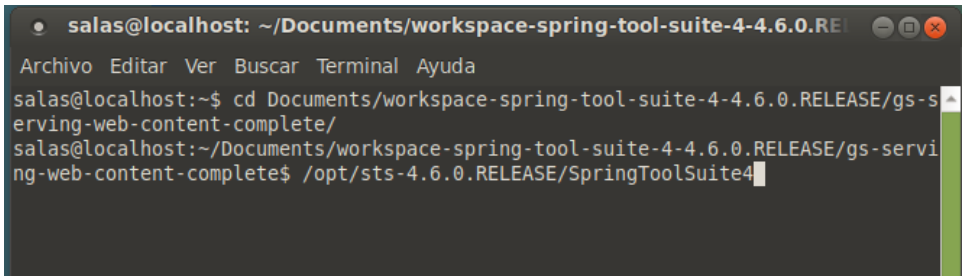

Figura 47: Iniciar Spring en Linux

2) Para crear un nuevo proyecto, pulsamos sobre File > New > Spring Starter Project. Nos aparecerá la siguiente pantalla:

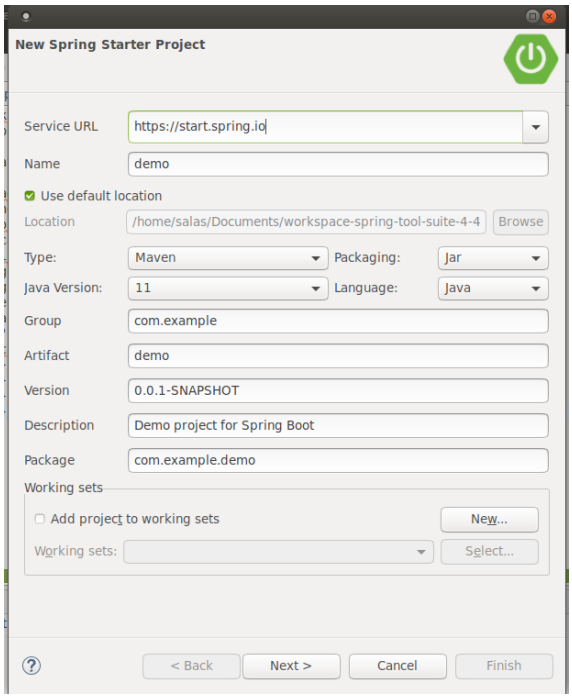

Figura 48: Creación nuevo proyecto - Spring

Como podemos observar, es en este paso donde elegiremos el nombre del proyecto, la tecnología Maven o Gradle, lenguaje de programación, empaquetado…

En este caso, la tecnología utilizada es **Maven.** ¿Y por qué?

Si buscamos información de Maven en algunos sitios como [24] o [25] todos coinciden en lo mismo: la gestión de librerías.

Cuando creamos un proyecto con Java hacemos uso de librerías. Normalmente, una librería depende de otras librerías y tenemos que buscar información sobre qué versión necesitan, cuáles son las

dependencias… etc. Antes de que existiese Maven podíamos preguntarnos cómo instalar una nueva librería, cómo podemos probarla… Pues bien, Maven resuelve este "problema" y nos agiliza el trabajo utilizando **artefactos**.

Podemos definir un artefacto como un "paquete" que incluye la librería, junto con su grupo de librerías de las que depende, versiones, otras dependencias… etc. Maven, se encargará de gestionar nuestro proyecto.

Al utilizar esta tecnología tan característica, se crea automáticamente una estructura de carpetas definida:

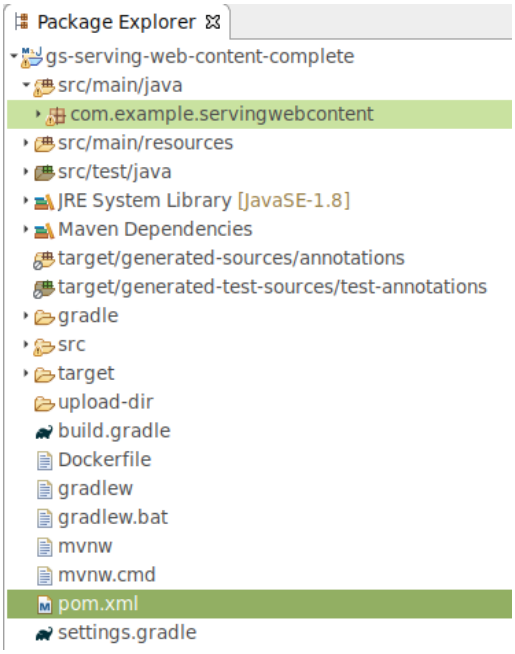

Figura 49: Estructura de directorios Maven

Todos los directorios tienen su propia función. Destacamos los más importantes dentro del proyecto:

- **src/main/java**: En este directorio se encontrarán las clases con extensión ".java"
- **src/main/resources**: Este directorio a su vez se divide en otros subdirectorios como vemos en la siguiente figura:

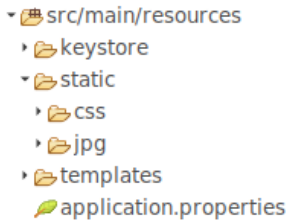

Figura 50: Contenido src/main/resources

En la carpeta "keystore" nos encontramos con el autocertificado que explicaremos en un apartado posterior. En la carpeta "static" estarán todas las imágenes utilizadas en el proyecto junto con el fichero .css que le dará formato a la aplicación web. En "templates" se han guardado todos los ficheros con extensión ".html" que forman la aplicación. Y por último, en el fichero **"application.properties"** vemos el siguiente contenido:

gs-serving-web-content-complete/pom.xml | application.properties

```
1  spring.servlet.multipart.max-file-size=128KB
2  spring.servlet.multipart.max-request-size=128KB
3  spring.batch.job.enabled = false
4
5  server.port: 4443
6  server.ssl.key-store: classpath:keystore/keystore.p12
7  server.ssl.key-store-password: certificado
8  server.ssl.keyStoreType: PKCS12
9  server.ssl.keyAlias: tomcat
10 server.ssl.enabled=true
11
```

Figura 51: Contenido "application.properties"

En las primeras líneas vemos algunas limitaciones como que solo podremos cargar en la página web ficheros de hasta 128KB, referido a la parte de la aplicación donde podremos cargar ficheros. A partir de la línea 5 vemos como hemos implementado el autocertificado en la aplicación web.

- El fichero **pom.xml** merece especial atención:

  Es muy importante para el buen funcionamiento de la aplicación que se complete el fichero pom.xml. Se encarga de gestionar el proyecto, ya que contiene sus dependencias, plugins… Si queremos ver a groso modo que contiene un proyecto, si revisamos este fichero podemos encontrar la información en la que está basada. Será en pom.xml donde definiremos los artefactos de los que hablamos en el apartado anterior.

  Cuando nos referimos a **dependencias**, son componentes que el proyecto necesita para que se ejecuten ciertas funciones. Se han ido actualizando durante la realización del proyecto, cuando se iban necesitando. Sabemos que se trata de una dependencia porque se escribe la etiqueta: **<dependency></dependency>.**

  En la siguiente captura podemos ver algunas de las dependencias del proyecto:

gs-serving-web-content-complete/pom.xml

```
41        <dependency>
42            <groupId>org.springframework.security</groupId>
43            <artifactId>spring-security-web</artifactId>
44
45        </dependency>
46        <dependency>
47            <groupId>org.springframework.security</groupId>
48            <artifactId>spring-security-config</artifactId>
49
50        </dependency>
51        <dependency>
52        <groupId>org.springframework.batch</groupId>
53        <artifactId>spring-batch-core</artifactId>
54    </dependency>
55        <dependency>
56            <groupId>org.thymeleaf.extras</groupId>
57            <artifactId>thymeleaf-extras-springsecurity4</artifactId>
58            <version>3.0.3.RELEASE</version>
59        </dependency>
60        <dependency>
61            <groupId>org.springframework.boot</groupId>
62            <artifactId>spring-boot-starter-security</artifactId>
63        </dependency>
64        <dependency>
65            <groupId>org.springframework.boot</groupId>
66            <artifactId>spring-boot-starter-thymeleaf</artifactId>
67        </dependency>
68        <dependency>
69            <groupId>org.springframework.boot</groupId>
70            <artifactId>spring-boot-starter-web</artifactId>
71        </dependency>
```

Figura 52: Fichero pom.xml - Spring

3) Una vez creado, Spring Boot nos ofrece una clase para arrancar directamente nuestra aplicación. Tendrá el mismo nombre que el proyecto. En nuestro caso:

Figura 53: Clase principal - Spring

Tenemos que destacar las directivas @SpringBootApplication y @EnableConfigurationProperties.

Con **@SpringBootApplication**, nuestro proyecto utilizará la **configuración automática**. Hoy en día es muy usada por la mayoría de los desarrolladores de Spring Boot. Con esa anotación, estamos indicando que nuestro trabajo utiliza **@Configuration, @EnableAutoConfiguration y @ComponentScan** con sus atributos predeterminados:

- @Configuration: Permite implementar clases de configuración adicionales o registrar beans adicionales. Al añadir esta anotación, la configuración se puede hacer en una clase Java y no en un fichero XML (como se suele hacer en otros proyectos que no usan Spring)
- @ComponentScan: Habilita el escaneo en nuestro proyecto. Trabaja junto a @Configuration, y le dice a Spring donde se encuentran los componentes que necesita nuestro proyecto.
- @EnableAutoConfiguration: Activa el mecanismo de configuración automática de Spring Boot, en función de las dependencias jar que se hayan ido agregando. Un ejemplo podría ser: Si HSQLDB (gestos de base de datos) se encuentra en su ruta de clase y no existe un bean de conexión de base de datos, la aplicación se encargará de configurar automáticamente una base de datos en la memoria.

En resumen, con esa anotación, se acelerará el desarrollo del proyecto. Podemos encontrar más información en la documentación oficial de Spring [26] y [27].

7.3 Spring Security Config

Para garantizar la seguridad de la aplicación, se ha utilizado la configuración **Spring Security Config**. De esta forma se ha podido restringir el acceso al servicio web mediante un formulario de inicio de sesión. En la siguiente captura podemos ver parte de la clase "SecurityConfig.java":

gs-serving-web-content-complete/pom.xml | application.properties | SecurityConfig.java

```
1  package com.example.servingwebcontent;
2
3  import org.springframework.beans.factory.annotation.Autowired;
10
11 /*
12  * Archivo de configuracion de Spring Security por el cual se determinan las paginas en las que se tiene acceso.
13  * Autenticacion y Autorizacion
14  */
15 @EnableWebSecurity
16 public class SecurityConfig extends WebSecurityConfigurerAdapter {
17
18
19     @Override
20     protected void configure(HttpSecurity http) throws Exception {
21         http
22             .csrf().disable()
23             .authorizeRequests()
24                 .antMatchers("/").permitAll()
25                 .antMatchers("/menu").hasRole("USER")
26                 .antMatchers("/mostrarFormulario").hasRole("USER")
27                 .antMatchers("/formularionuevo").hasRole("USER")
28                 .antMatchers("/recomendacion").hasRole("USER")
29                 .antMatchers("/desbloquearURLs").hasRole("USER")
30                 .antMatchers("/watchlist").hasRole("USER")
31                 .and()
32             .formLogin()
33                 .loginPage("/iniciarSesion")
34             .and().exceptionHandling().accessDeniedPage("/accessdenied")
35             .and()
36                 .logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout")).logoutSuccessUrl("/iniciarSesion")
37                 .deleteCookies("JSESSIONID");
38
39
40
41     }
42
```

Figura 54: Fichero SecurityConfig.java - Spring

Configuramos qué pantallas estarán bloqueadas hasta que no se inicie sesión y cuáles no. Para ello utilizaremos la sentencia ".antMatchers()". Todas aquellas pantallas (/menu, /mostrarFormulario…) que tengan la opción ".hasRole" significa que necesitan que el usuario se haya autenticado previamente para poder visualizarla. Si el usuario no ha iniciado sesión correctamente, la propia aplicación enviará al usuario a la pantalla /iniciarSesion:

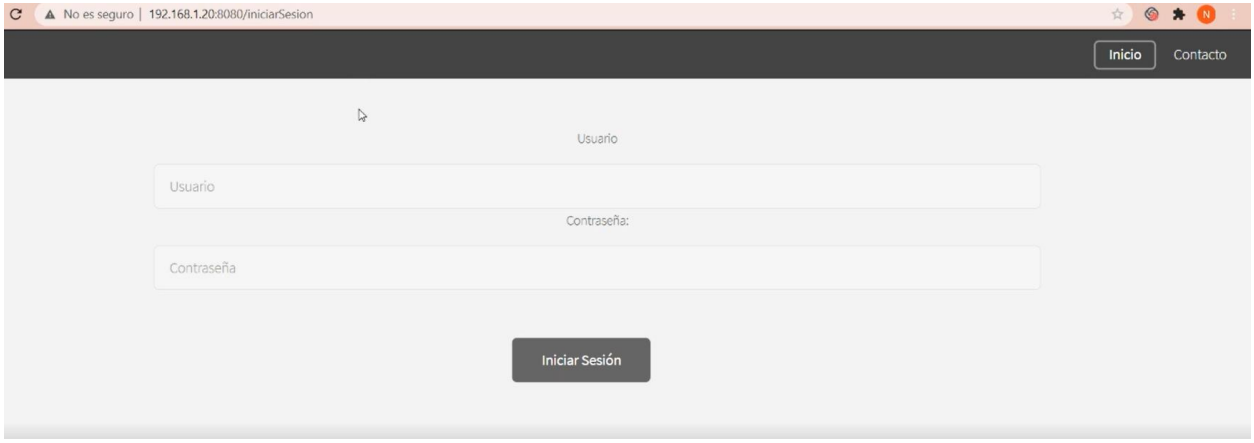

Figura 55: Pantallar iniciar sesión - Spring

Sin embargo, si nos fijamos en la línea 24 de la Figura 29, comprobamos que efectivamente para acceder a la pantalla principal (la primera que aparece al entrar en la aplicación web), no se necesita autenticación.

El usuario "USER" se encuentra registrado en la aplicación para que sea el único que pueda acceder.

Podemos encontrar más información sobre este tipo de configuración en [28].

7.4 Spring Web MVC

Como ya se ha mencionado, la principal tarea de la aplicación web es bloquear el acceso a las páginas web que deseemos. Para ello, se ha implementado siguiendo el **Modelo-Vista-Controlador de Spring Web.**

Cada **controlador es el que recibe las peticiones del cliente** (capturándolas a través de ciertas anotaciones) y es el que se encarga de devolverle la vista al usuario. Las anotaciones a destacar son:

- **@Controller:** Indica que la clase desarrollada es un controlador. Es decir, que se encargará de recibir peticiones HTTP de los clientes y actuará en consecuencia. Para más información, se encuentra detallado en la página principal de Spring [29].
- **@RequestMapping, @PostMapping, @GetMapping**: Estas tres anotaciones se utilizan para indicar a Spring que el método anotado es el que debe procesar la petición recibida. Así, si anotamos un método en particular con @GetMapping("/urlconcreta"), se ejecutará este método cada vez que nos llegue la petición. De esta forma, Spring nos permite olvidarnos de la configuración y centrarnos en la programación. [30] , [31], [32]

Así, la única interacción que existe es con el controlador, quién se encarga de procesar todo, y, en caso de acceder a datos, es él mismo quién lo maneja. El funcionamiento del modelo-vista-controlador se estudia en la asignatura "Ingeniería de Software", tal y como podemos leer en la siguiente figura:

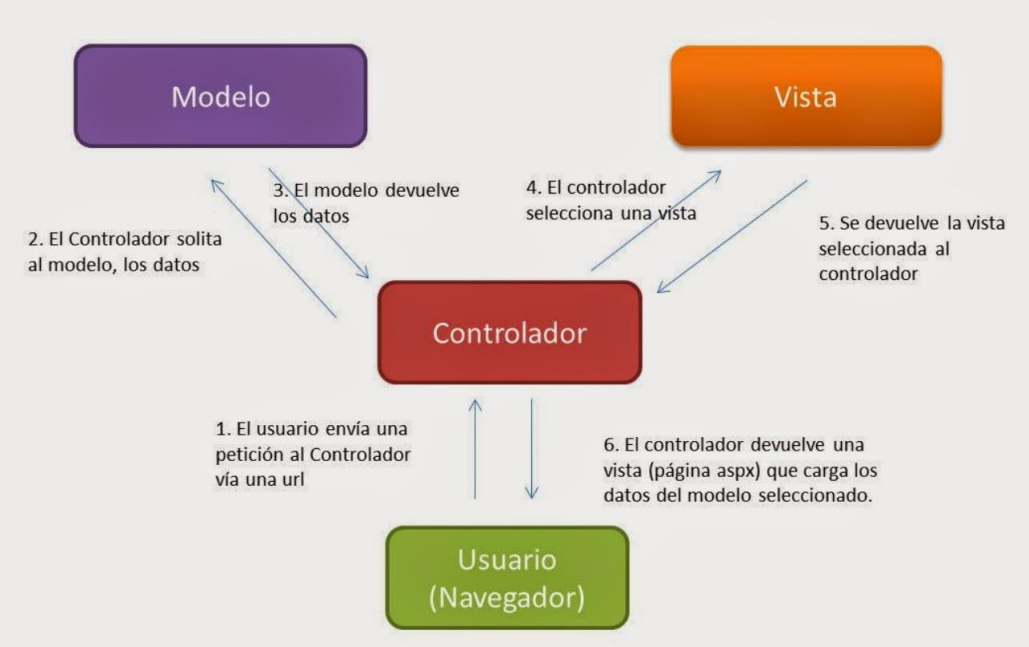

Figura 56: Procedimiento MVC

Aplicándolo a nuestra aplicación web en Spring, podemos seguir el siguiente proceso:

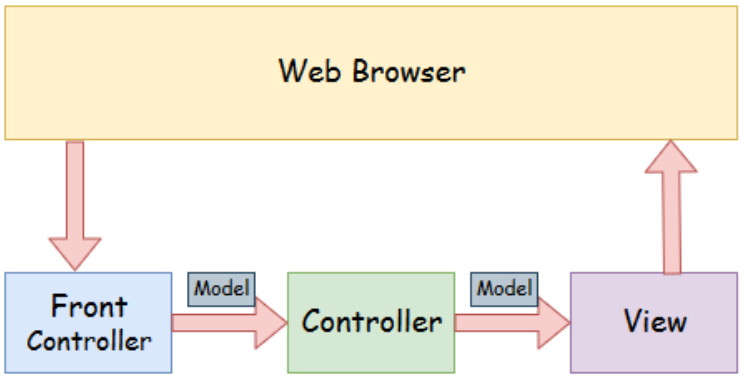

Figura 57: Modelo-Vista-Controlador Spring Web

- Model/Modelo: Será la clase que contiene los datos de la aplicación. En este caso al ser una aplicación sin base de datos, nadie actuará de modelo.
- Controller/Controlador: Es el que contiene la lógica de la aplicación. Son las clases que contienen la palabra "Controller"
- View/Vista: Representará las ventanas de la aplicación.
- Front Controller/Controlador Frontal: En Spring, es la clase DispatcherServlet la que se encarga de manejar el flujo de la aplicación MVC. Será quien reciba las solicitudes y les asigne los recursos correctos (controladores, modelos y vistas).

El proyecto se compone de las siguientes clases:

Estos controladores, con sus métodos debidamente desarrollados, se encargan de escribir las páginas solicitadas en nuestra lista de URLs bloqueadas.

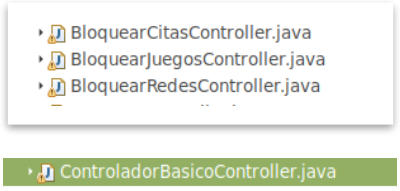

Figura 58: Controladores bloquear URLs - Spring

Siguiendo el modelo-vista controlador, los controladores irán capturando todas las peticiones del usuario. Los ficheros con extensión **.html** serán los que tengan el papel de **vista**, ya que le irán ofreciendo al usuario las diferentes pantallas en función de su petición. Como explicamos en los objetivos del proyecto, una de nuestras prioridades era desarrollar una distribución ligera. Si hubiésemos implementado una base de datos en el proyecto, hubiese ocupado más memoria de la necesaria y no hubiésemos podido desplegar la aplicación en un equipo antiguo. Por tanto, todos los datos se encuentran en ficheros de texto en la propia aplicación. En el caso de estos controladores, su funcionalidad es bloquear ciertas páginas web, por lo que hemos creado un **fichero** "paginasbloqueadas.txt" donde iremos **almacenando todas las URL**s que el usuario quiera bloquear para posteriormente gestionar el proxy Squid con el contenido de este. En la siguiente figura vemos el proceso descrito:

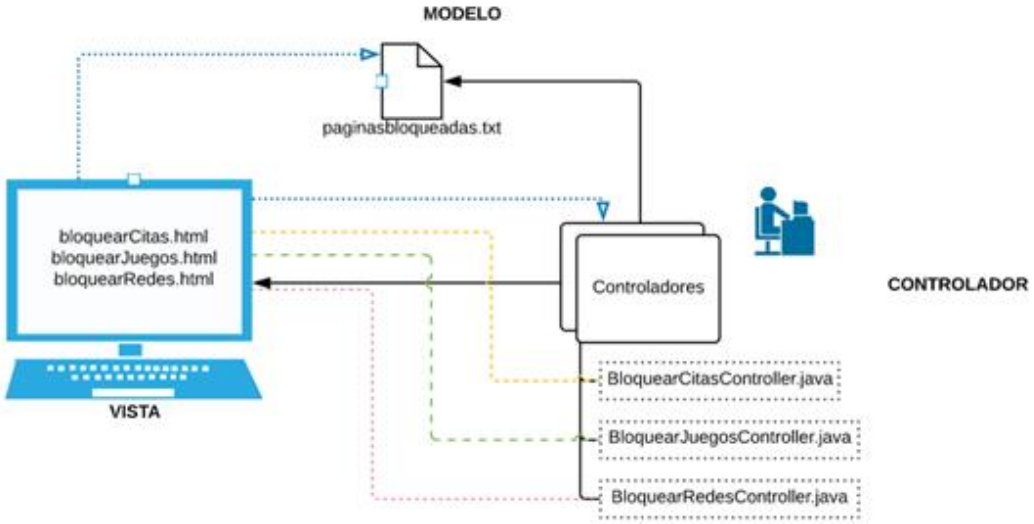

Figura 59: MVC aplicado al proyecto

Por el contrario, la clase DesbloqueoController.java se encarga de eliminar de nuestra lista la página solicitada. El funcionamiento es parecido a los controladores de bloqueo, con la clara diferencia que ahora en vez de escribir en el fichero, lo que hace es eliminar del fichero la URL solicitada por el usuario.

Figura 60: Controlador desbloquear URLs - Spring

Este tipo de controladores se utilizan para mostrar por pantalla las diferentes ventanas de nuestra aplicación. En el caso de IndexController.java, mostrará la ventana principal. MenuController.java mostrará la ventana que aparece justo al autenticarse, con todas las funciones que ofrece la aplicación. RecomendacionController.java mostrará una ventana donde el usuario podrá ver las URLs que los administradores de la aplicación recomiendan bloquear. Por último, ContactoController.java mostrará una ventana donde aparecerá información relacionada con el creador de la aplicación web.

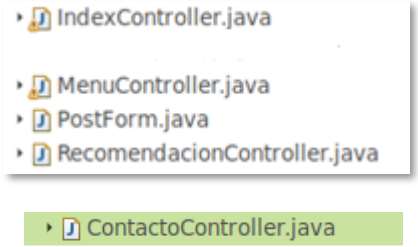

Figura 61: Controladores pantallas – Spring

Gracias a estas dos clases, la aplicación ofrece la posibilidad de ver las páginas web que han sido bloqueadas por el usuario.

Figura 62: Controlador y clase listar URLs – Spring

En las opciones anteriores, el usuario tenía que ir introduciendo una a una las URLs que quería bloquear. Para facilitarle la tarea, la aplicación dispone de una opción para cargar un fichero txt con todas las páginas que prohíbe el acceso. Gracias a las clases FileSystemStorageService.java y FileUploadController.java se ha podido implementar.

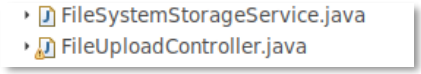

Figura 63: Clases que gestionan la carga de txt – Spring

Además de las propiedades indicadas en el fichero "application.properties" en la Figura 25, también implementamos las siguientes clases para controlar si el fichero no se ha cargado correctamente o si estuviese vacío.

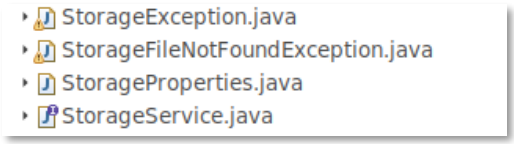

Figura 64: Clases que gestionan el almacenamiento de txt - Spring

En definitiva, lo que hacen estas clases es leer el fichero que el usuario carga en la aplicación y copiar el contenido en el fichero donde almacenamos todas las páginas web a bloquear. De esta manera, el proxy no tendrá que bloquear más de un fichero dinámicamente.

Para garantizar la seguridad de la aplicación, únicamente podrá iniciar sesión aquellos usuarios debidamente registrados. Con las dos clases de a continuación, aseguramos la autenticación:

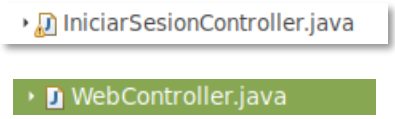

Figura 65: Controladores autenticación de usuarios - Spring

El escenario de la Figura 33 podría ser aplicado para todos los controladores descritos, con la siguiente relación:

MODELO	VISTA	CONTROLADOR
Fichero de texto	bloquearCitas.html	BloquearCitasController.java
Fichero de texto	bloquearJuegos.html	BloquearJuegosController.java
Fichero de texto	bloquearRedes.html	BloquearRedesController.java
Fichero de texto	desbloqueo.html	DesbloqueoController.java
Fichero de texto	formulario.html	ControladorBasicoController.java
-	/	IndexController.java
-	iniciarSesion.html	IniciarSesionController.java
-	menu.html	MenuController.java
-	recomendacion.html	RecomendacionController.java
-	contacto.html	ContactoController.java
Fichero de texto	watchlist.html	WatchListController.java
Fichero de texto	formularionuevo.html	FileUploadController.java

Tabla 1: Correspondencia Modelo-Vista-Controlador

En resumen, podríamos decir que gracias a la aplicación web realizada con la tecnología Spring, podemos configurar el proxy Squid sin que el usuario final se de cuenta.

7.5 Comunicación segura a través de HTTPs

Cuando en el apartado 3.2 vimos los protocolos, mencionamos la importancia de ofrecerle seguridad al usuario que utilice la aplicación web. Por defecto, las aplicaciones web creadas mediante Spring Boot utilizan HTTP en el puerto 8080. Para que funcionen mediante HTTPs, necesitamos crear un certificado SSL/TLS y luego añadirlo en la aplicación con el fichero application.properties. Mediante el certificado, se garantiza que la aplicación es confiable. En el apartado 3.2 también se mencionaron los dos tipos de certificación y el que se iba a utilizar en el proyecto: **el autocertificado.**

Como la aplicación web se desarrolla en la máquina virtual de Linux, generaremos ahí nuestro certificado. Para ello, ejecutamos desde el terminal el siguiente comando:

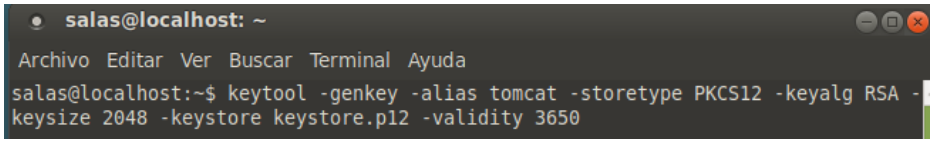

Figura 66: Comando Linux creación autocertificado

Después simplemente rellenamos los datos personales de la entidad que se quiera autenticar. En este caso, se rellenan los datos de la desarrolladora de la aplicación:

```
Introduzca la contraseña del almacén de claves:
La contraseña del almacén de claves es demasiado corta, debe tener al menos 6 ca
racteres
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
  [Unknown]:  Natalia Sanchez Portillo
¿Cuál es el nombre de su unidad de organización?
  [Unknown]:  ETSI
¿Cuál es el nombre de su organización?
  [Unknown]:  Ingenieria
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:  Sevilla
¿Cuál es el nombre de su estado o provincia?
  [Unknown]:  Sevilla
¿Cuál es el código de país de dos letras de la unidad?
  [Unknown]:  ES
¿Es correcto CN=Natalia Sanchez Portillo, OU=ETSI, O=Ingenieria, L=Sevilla, ST=S
evilla, C=ES?
  [no]:  si

salas@localhost:~$
```

Figura 67: Datos a rellenar autocertificado

Comprobamos que efectivamente se ha creado el autocertificado en el directorio donde ejecutamos el comando y que es necesaria una contraseña para abrirlo:

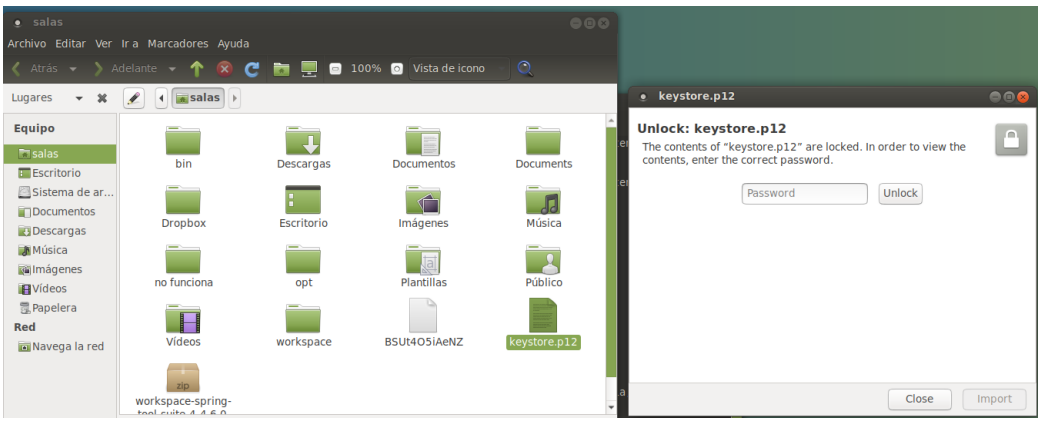

Figura 68: Autocertificado creado (1)

Efectivamente aparecen los datos que se introdujeron mediante el terminal. Si desplegamos “Details” aparecerá la información acerca de la clave pública, cifrado…

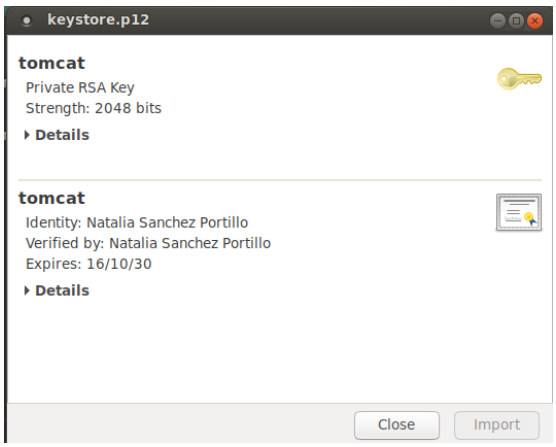

Figura 69: Autocertificado creado (2)

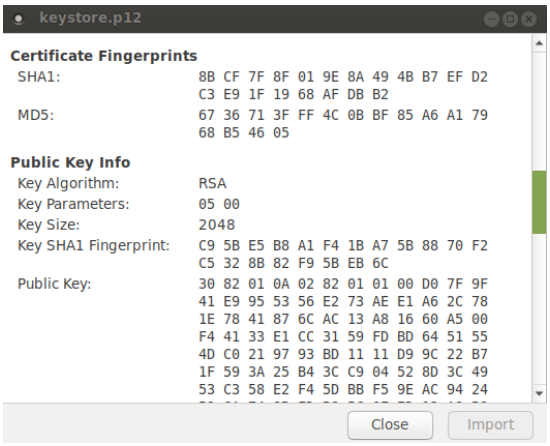

Figura 70: Autocertificado creado (3)

Una vez creado el certificado, ya solo faltaría añadirlo en nuestra aplicación. En la **Figura 24** comprobamos que se ha creado una carpeta específica para guardar el certificado, **"keystore"** . En la **Figura 25** podemos ver cómo se ha implementado el certificado en la aplicación.

Ya únicamente faltaría cargarlo en el navegador de confianza. Cuando accedemos a la aplicación web con la URL https://IP-Servidor:4443 podemos encontrar el siguiente cuadro informativo:

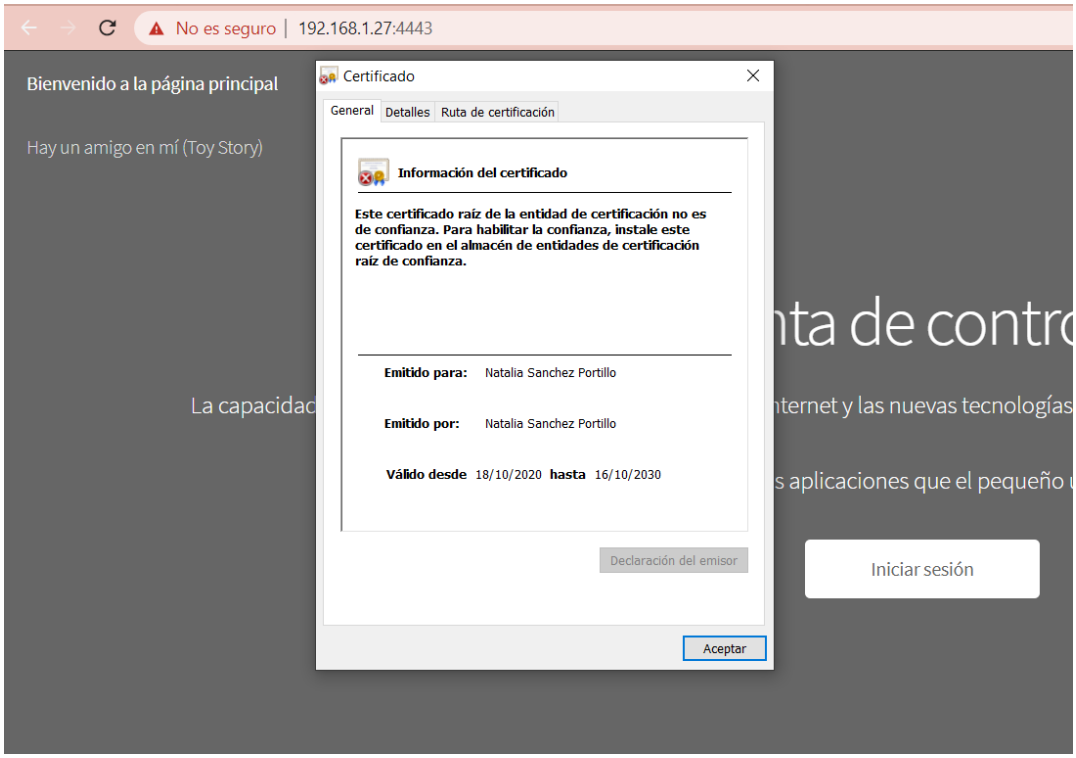

Figura 71: Certificado en el navegador

Al ser un certificado autofirmado, es importante entender que no estamos consiguiendo la autenticación del servidor: como estudiamos en la asignatura de Seguridad, para ello el certificado debería estar expedido y firmado por una autoridad de certificación (AC) que reconozcamos, como por ejemplo la FNMT o el ANCERT. Sin embargo, lo que sí aseguramos es la confidencialidad entre emisor y receptor: un atacante no podrá interceptar el mensaje y descifrar su contenido.

7.6 Servidor Web

Una pregunta que nos puede surgir es: ¿Y el servidor web donde está?

Otro de los motivos por los que se elegio Spring Boot fue porque incorpora servidores web como Apache o Tomcat. Por tanto, no fue necesario configurar un servidor web como tal, si no que solo tuvimos que adaptar la aplicación para que utilizase el que integra Spring Boot. A través de líneas de configuración en el fichero application.properties, se puede especificar el puerto en el que escucha el servidor, las opciones…

8 CONCLUSIONES Y LÍNEAS FUTURAS

8.1 Conclusiones personales

Ahora puedo echar la vista atrás y comprobar que efectivamente se han cumplido los objetivos mencionados al principio de la memoria. El resultado del proyecto ha sido muy satisfactorio tanto a nivel personal como profesional.

Cuando decidí matricularme del TFG, no sabía de qué me gustaría hacerlo. Pero siempre tuve claro que quería que tuviese un resultado muy visual. Que pudiese serle de utilidad tanto a las personas relacionadas con el mundo tecnológico, como a las que no. Y así ha sido. Además, al poder personalizar el estilo de la aplicación web a mi gusto, he podido dejar una pequeña parte de quien soy ahora entre esas líneas de código.

Ha habido momentos de frustración y de duda. Pero ahora que puedo disfrutar del resultado, me doy cuenta de cuánto he aprendido estos meses. Las habilidades que he adquirido después de la realización de este proyecto y todo lo que puedo seguir creciendo después de esto. Aún sin haber antes creado nada propio, con la base y formación que nos aporta el Grado de Ingeniería y junto con los consejos y guía de mi tutor, se han conseguido los objetivos.

8.2 Conclusiones técnicas y líneas futuras

Han sido muchos los retos técnicos que han ido surgiendo durante el desarrollo del Proyecto. Aún conociendo los conceptos básicos de Linux y Java, no ha sido fácil el desarrollo de la aplicación y la implementación de la distribución.

En primer lugar, no es usual trabajar con una distribución sin interfaz gráfica. **Todo ha sido configurado y personalizado mediante el terminal** y el repositorio que ofrece Alpine Linux. Esta situación de primeras no era muy intuitiva y pasaron muchos días hasta que pude familiarizarme con los nuevos comandos y directorios.

Tampoco nunca antes había trabajado con un proxy, y aunque sabia superficialmente qué era y para qué servía, me faltaban algunos conceptos como: orden de las reglas de filtrado, existencia de los diferentes ficheros de configuración, opciones…

Aunque en la asignatura FAST de 2º de GITT trabajamos con aplicaciones web, nunca habia oído hablar de Spring Framework. Fue empezar desde cero. Hasta que no conseguí la destreza y las habilidades necesarias para este entorno, no pude comenzar a programar.

Tampoco fue sencillo dockerizar la aplicación web para poder desplegarla en la minidistribución. Fue necesario una gran investigación acerca de su nube, comandos y su independencia frente a la máquina donde fuese implementado.

Aún siendo el resultado muy satisfactorio, propongo algunas mejoras para futuras versiones:

- Implementar una base de datos donde guardar los dominios de las páginas web en vez de guardarse en un fichero txt. Esto implicaría un aumento del almacenamiento de la distribución, pero podría plantearse para equipos más preparados.
- Implementar un registro de usuarios en tiempo de ejecución.
- Utilizar Docker-compose para añadirle más funcionalidades a la aplicación web.

ANEXO A: INSTALACIÓN Y CONFIGURACIÓN ALPINE LINUX

Explicaremos detalladamente cada paso a seguir para instalar la distribución:

1. Descargar la imagen ISO

Descargamos desde la página oficial [35] la versión Estándar x86_64. En nuestro caso hemos elegido esa versión porque el procesador del equipo donde va a ser instalado la distribución es de 64 bits. Al querer que ocupe poco espacio, no elegimos la version Extended, ya que únicamente queremos que contenga lo básico. No podemos olvidar que cualquier paquete que necesitemos podremos instalarlo desde el propio repositorio de la distribución.

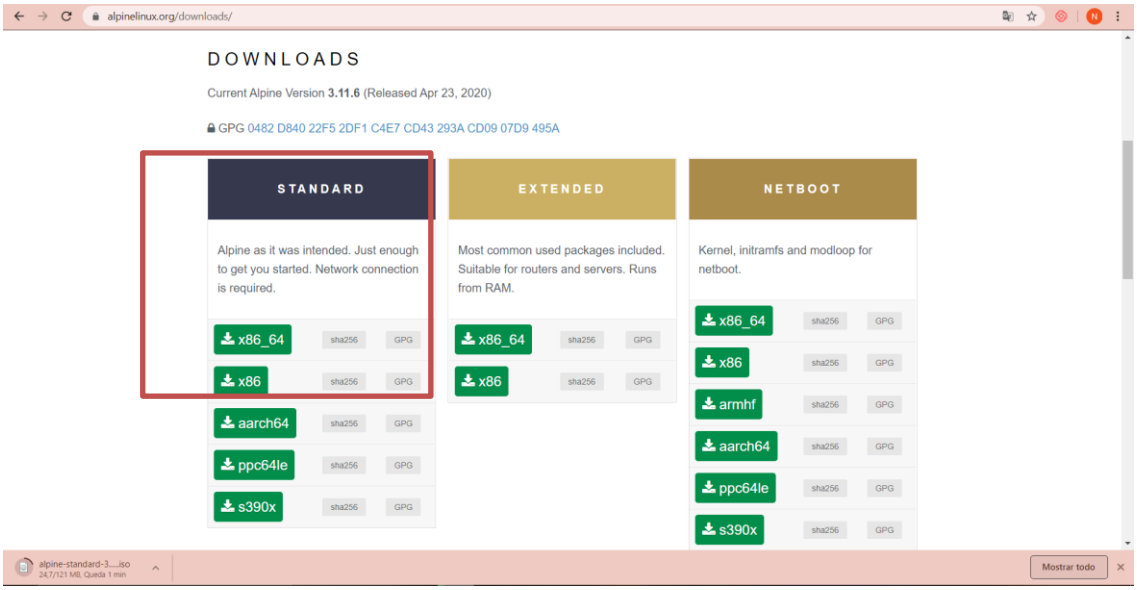

Figura 72: Descarga imagen ISO Alpine Linux

2. Instalamos la distribución en nuestro USB

Una vez tenemos descargada la imagen ISO de nuestra distribución, el siguiente paso será instalar Alpine Linux en nuestro USB. Para ello utilizaremos el programa **UNetbootin**.

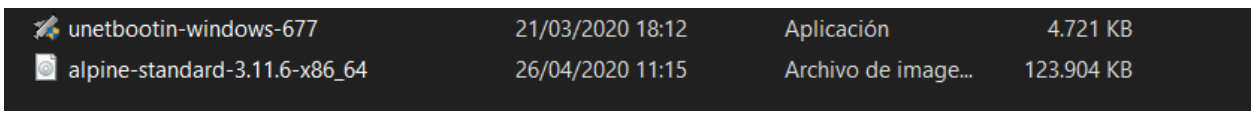

Figura 73: Programa e imagen para instalar en el USB

Recordemos que otro de nuestros objetivos es salvar equipos antiguos de forma que puedan funcionar sin disco, por tanto, arrancará la distribución gracias al USB. UNetbootin es un programa que nos permite crear unidades USB Live de cualquier distribución de Linux únicamente necesitando su imagen ISO.

Antes de nada, necesitamos formatear nuestro USB para que quede totalmente vacío y disponible para su nuevo funcionamiento.

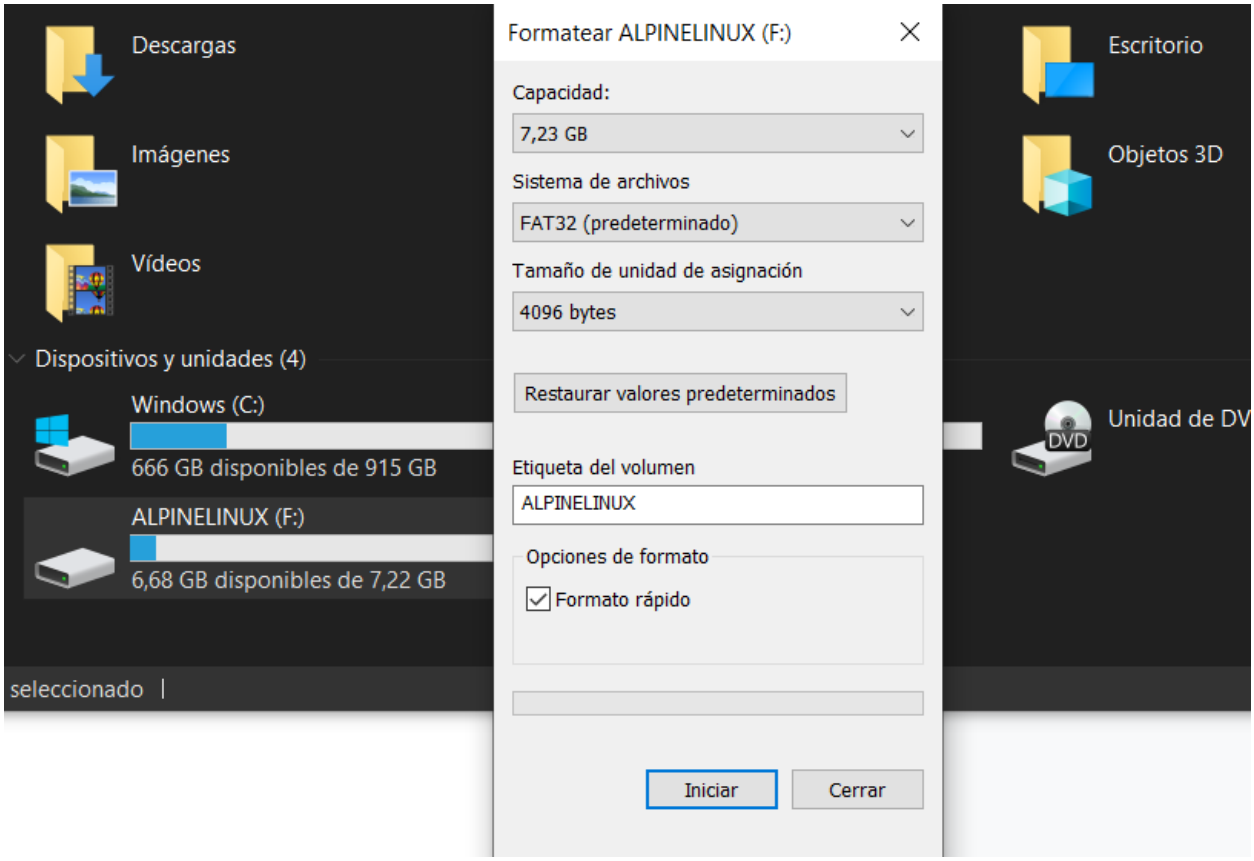

Figura 74: Formateo USB

Una vez formateado, ejecutamos el programa UNetbootin y cargamos la imagen ISO de Alpine Linux elegida:

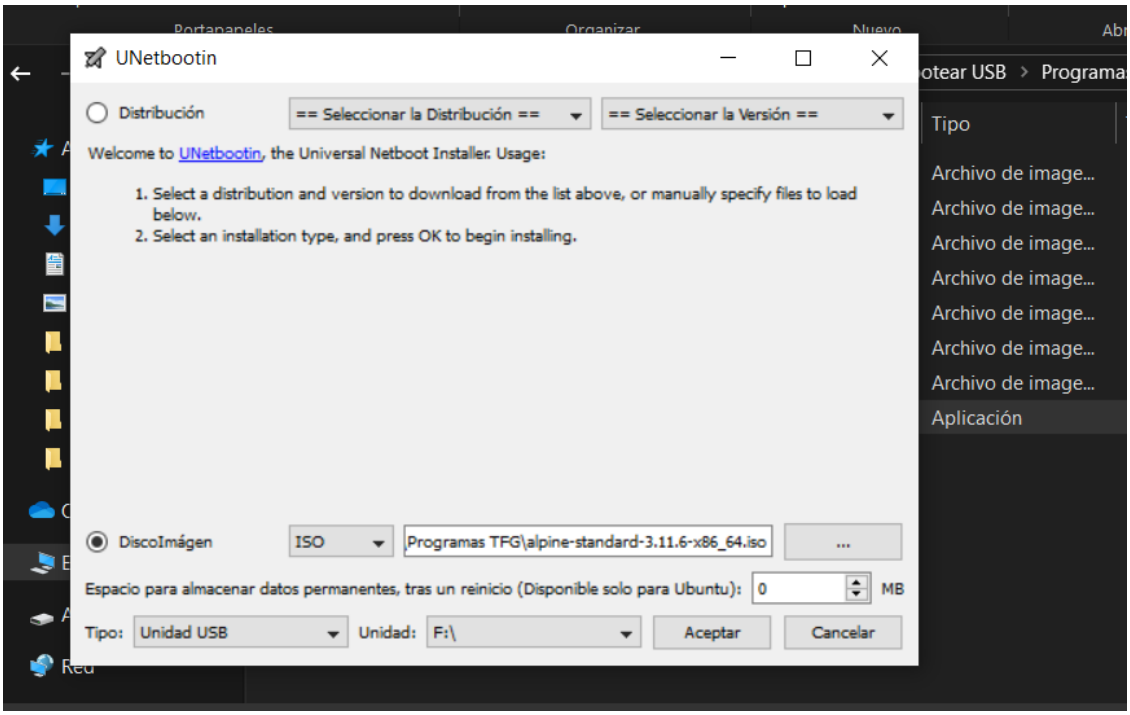

Figura 75: Creación USB Booteable

En la siguiente imagen podemos observar las fases por las que va pasando la imagen ISO hasta quedar instalada sin ningún error en nuestro USB:

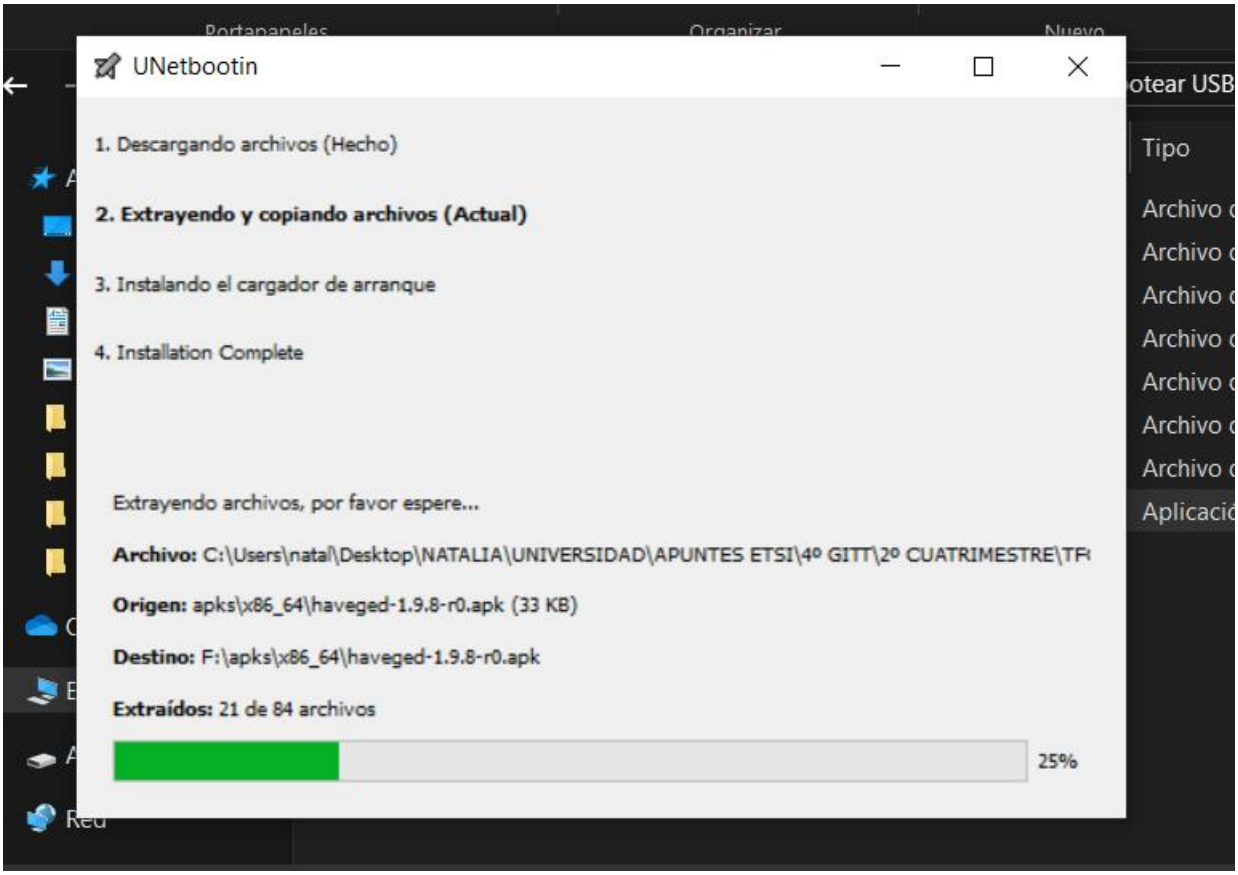

Figura 76: UNetbootin funcionamiento

Una vez finalizada la conversión, podemos comprobar que efectivamente la distribución ocupa muy poco espacio en la memoria.

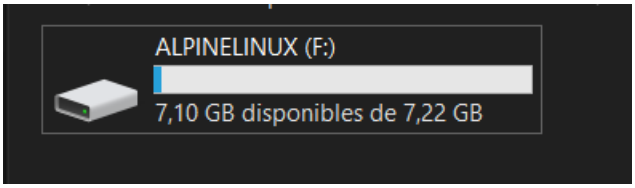

Figura 77: Capacidad distribución en USB

Llegados a este punto, podemos pasar a ejecutar nuestra nueva distribución en el equipo que funcionará de servidor.

3. Proceso de arranque e instalación

Una vez instalada la distribución, al arrancar el equipo nos aparecerá la siguiente pantalla:

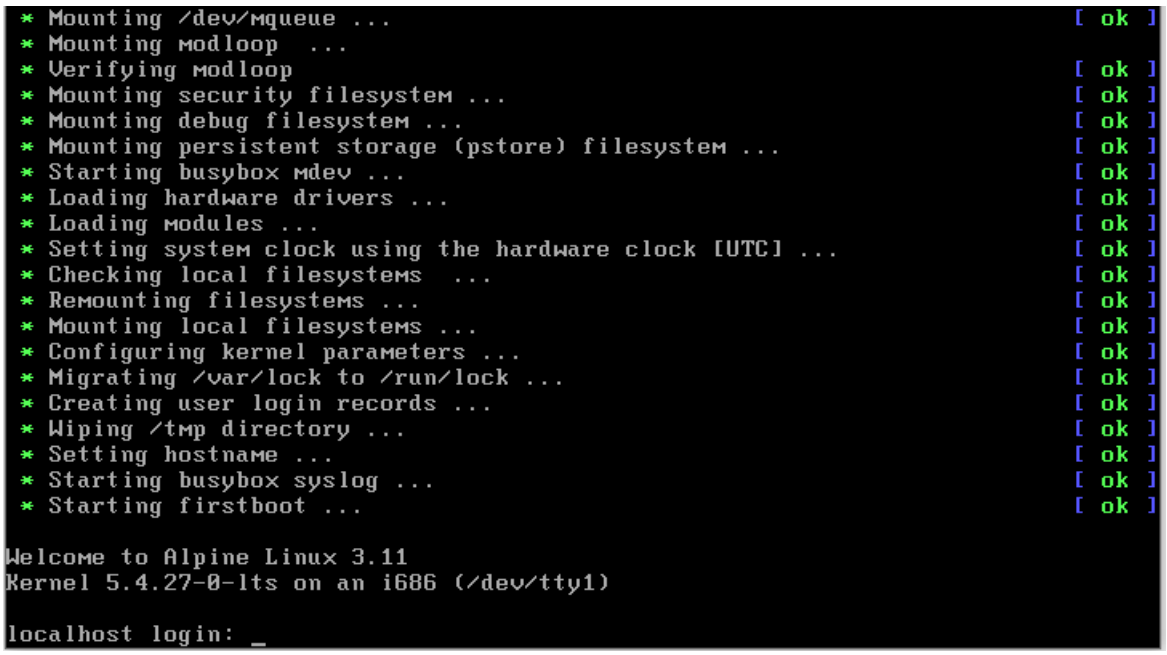

Figura 78: Pantalla inicial Alpine Linux

Inicialmente no tenemos ningún usuario, por lo que iniciamos sesión como usuario root y sin ninguna contraseña. A continuación, ejecutamos el comando setup-alpine y podremos configurar algunas de las características de nuestra distribución:

- Elegimos la distribución del teclado: En nuestro caso elegiremos “es” (Español)
- Nos ofrecen elegir variante del teclado: También elegiremos “es” (Español)
- Configuramos el nombre del host: En nuestro caso lo llamaremos “tfg-natalia”
- Configuración de las tarjetas de red: Seleccionamos la red con la que vayamos a trabajar así como su contraseña.

- Elegimos nombre de dominio: Al no necesitar ninguno específico, pulsamos enter.
- Servidores DNS: Seleccionamos el que aparece por defecto, por lo que pulsamos enter.
- Cambiar la contraseña de root: Elegimos una contraseña
- Zona horaria: En nuestro caso elegimos Europa central.
- Selección de proxy: Al no necesitar ninguno en concreto de la distribución escribimos none.
- Servidor ssh: Al pulsar enter seleccionamos ssh para administrar el equipo de forma repota.
- Disco de configuración: Como en nuestro caso arrancaremos la distribución mediante USB, seleccionaremos sdb.

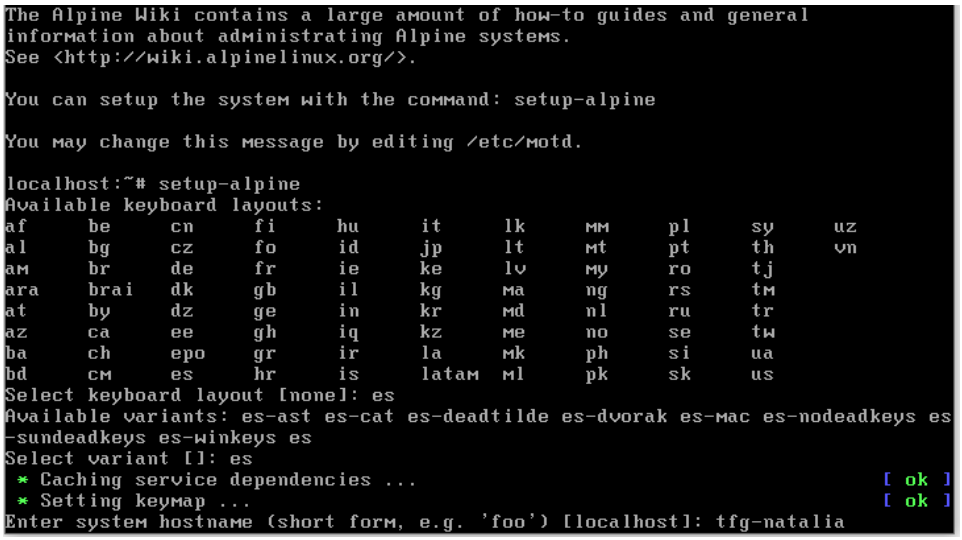

Figura 79: Configuración Alpine Linux

Una vez rellenado todos los campos, comprobamos la conectividad a Internet haciendo algunos pings:

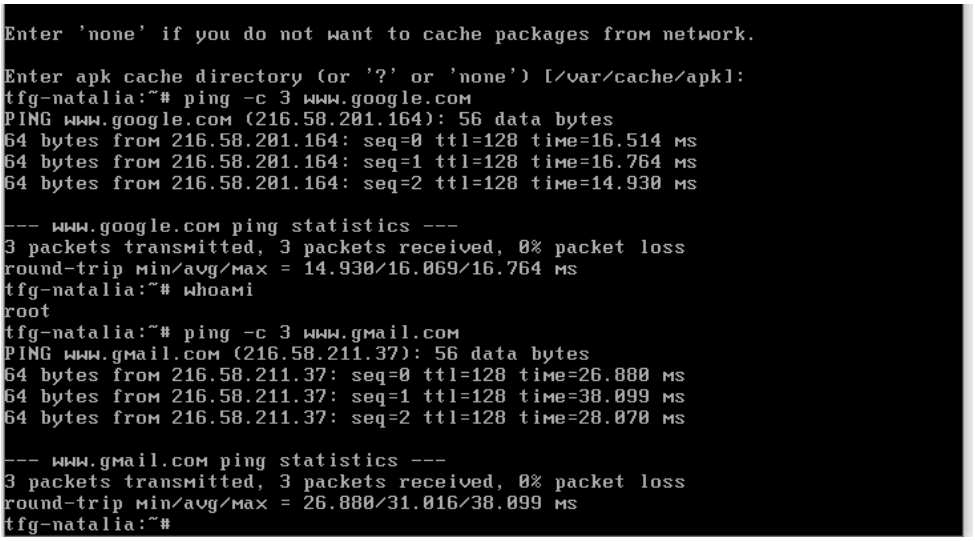

Figura 80: Prueba conectividad Alpine Linux

Efectivamente, nuestra distribución funciona y ha sido configurada correctamente.

ANEXO B: INSTALACIÓN Y CONFIGURACIÓN TECNOLOGÍAS UTILIZADAS

Internet es mucho más que una tecnología. Es un medio de comunicación, de interacción y de organización social.

-Manuel Castells-

En este apartado explicaremos detalladamente cómo instalar las tecnologías usadas en el proyecto y su configuración.

Tecnologías

Spring Tool Suite

Para instalar el entorno Spring Tool Suite (STS) en Ubuntu, antes necesitamos comprobar que se encuentra Java instalado en nuestro equipo.

>> **java -version**

Una vez verificado, instalamos Maven, necesario para la creación de la aplicación en Spring Too Suite:

>> **sudo apt install maven**

Si no ocurre ningún error, pasamos a instalar la última versión de STS. Para ello, desde su página oficial [36] pulsamos sobre el botón "LINUX 64-bit":

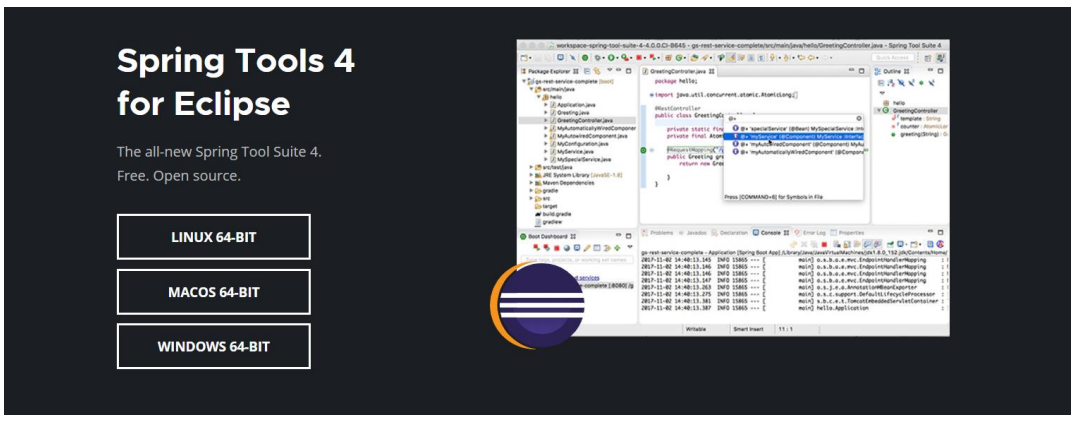

Figura 81: Descargar Spring Framework

Se descargará el archivo comprimido y debemos descomprimirlo en el directorio /opt:

>> sudo mv spring-tool-suite-4-4.6.0.RELEASE-e4.15.0-linux.gtk.x86_64.tar.gz /opt
>> cd /opt
>> sudo tar xfvz spring-tool-suite-4-4.6.0.RELEASE-e4.15.0-linux.gtk.x86_64.tar.gz

Una vez instalado el entorno, para iniciarlo escribimos en el terminal los comandos que aparecen en la siguiente captura:

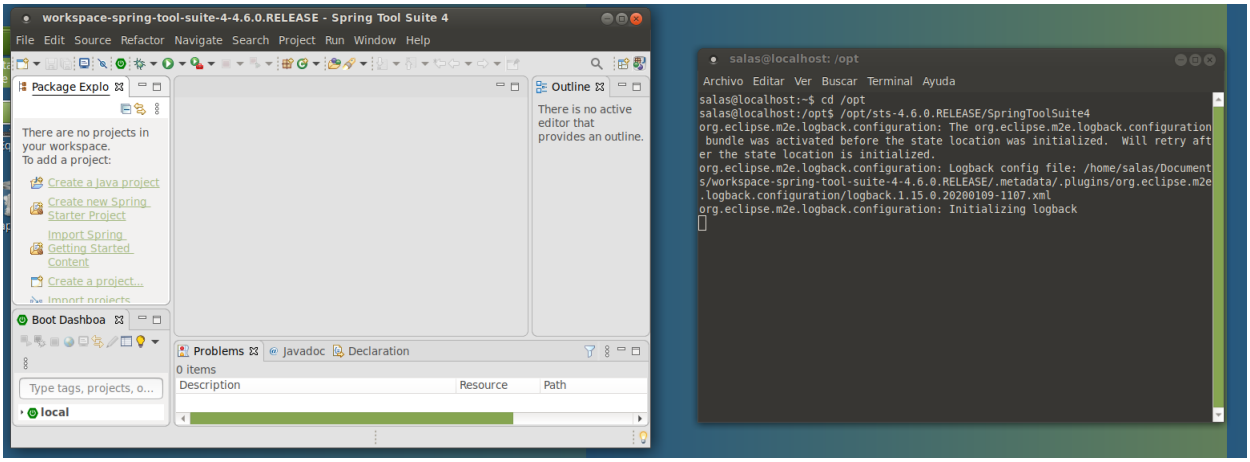

Figura 82: Espacio de trabajo Spring

Docker

Necesitaremos instalar Docker en el equipo con **Ubuntu** y en el equipo con **Alpine Linux.**

Para Ubuntu:

Utilizando [37]:

En primer lugar actualizamos la lista de paquetes existentes por si fuese necesario:

>> sudo apt update

Para que apt pueda utilizar HTTPs instalamos la siguiente dependencia:

>> sudo apt install apt-transport-https ca-certificates curl software-properties-common

Añadimos la clave de Docker:

>> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

Añadimos el repositorio de Docker:

>> sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

Volvemos a actualizar la lista de paquetes:

>> sudo apt update

Por último instalamos Docker:

>> sudo apt install docker-ce

Para Alpine Linux:

Utilizando [38] :

Gracias a su repositorio, ejecutamos:

>> apk add docker

De esta forma obtenemos Docker en ambos equipos.

Squid

Para instalar Squid en la distribución Alpine Linux, ejecutamos el siguiente comando:

```
tfg-natalia:~# apk add squid
(1/8) Installing popt (1.16-r7)
(2/8) Installing logrotate (3.15.1-r0)
(3/8) Installing logrotate-openrc (3.15.1-r0)
(4/8) Installing db (5.3.28-r1)
(5/8) Installing libgcc (9.2.0-r4)
(6/8) Installing libltdl (2.4.6-r7)
(7/8) Installing libstdc++ (9.2.0-r4)
(8/8) Installing squid (4.10-r0)
Executing squid-4.10-r0.pre-install
Executing busybox-1.31.1-r9.trigger
OK: 26 MiB in 40 packages
tfg-natalia:~# _
```

Figura 83: Instalación Squid en Alpine Linux

Ejecutamos "**apk add squid-doc**" y "**apk add squid-lang-es**" para utilizar su documentación y poder leerla en español. Si queremos ver la lista completa de idiomas disponibles, ejecutamos el siguiente comando:

```
tfg-natalia:~# ls /usr/share/squid/errors
COPYRIGHT    en-gb        en-tt        es-bo        es-gt        es-py
TRANSLATORS  en-ie        en-uk        es-cl        es-hn        es-sv
en           en-in        en-us        es-co        es-mx        es-us
en-au        en-jm        en-za        es-cr        es-ni        es-uy
en-bz        en-nz        en-zw        es-do        es-pa        es-ve
en-ca        en-ph        es           es-ec        es-pe        es-xl
en-cn        en-sg        es-ar        es-es        es-pr        templates
tfg-natalia:~#
```

Figura 84: Idiomas disponibles en Squid

Podemos configurar el mensaje de error que nos muestre Squid por pantalla cada vez que nos bloquea un acceso. En la siguiente captura vemos todos los tipos de errores que podemos personalizar:

```
en-cn        en-sg        es-ar        es-es        es-pr        templates
tfg-natalia:~# ls /usr/share/squid/errors/es-es
ERR_ACCESS_DENIED            ERR_ICAP_FAILURE
ERR_ACL_TIME_QUOTA_EXCEEDED  ERR_INVALID_REQ
ERR_AGENT_CONFIGURE          ERR_INVALID_RESP
ERR_AGENT_WPAD               ERR_INVALID_URL
ERR_CACHE_ACCESS_DENIED      ERR_LIFETIME_EXP
ERR_CACHE_MGR_ACCESS_DENIED  ERR_NO_RELAY
ERR_CANNOT_FORWARD           ERR_ONLY_IF_CACHED_MISS
ERR_CONFLICT_HOST            ERR_PRECONDITION_FAILED
ERR_CONNECT_FAIL             ERR_PROTOCOL_UNKNOWN
ERR_DIR_LISTING              ERR_READ_ERROR
ERR_DNS_FAIL                 ERR_READ_TIMEOUT
ERR_ESI                      ERR_SECURE_CONNECT_FAIL
ERR_FORWARDING_DENIED        ERR_SHUTTING_DOWN
ERR_FTP_DISABLED             ERR_SOCKET_FAILURE
ERR_FTP_FAILURE              ERR_TOO_BIG
ERR_FTP_FORBIDDEN            ERR_UNSUP_HTTPVERSION
ERR_FTP_NOT_FOUND            ERR_UNSUP_REQ
ERR_FTP_PUT_CREATED          ERR_URN_RESOLVE
ERR_FTP_PUT_ERROR            ERR_WRITE_ERROR
ERR_FTP_PUT_MODIFIED         ERR_ZERO_SIZE_OBJECT
ERR_FTP_UNAVAILABLE          error-details.txt
ERR_GATEWAY_FAILURE
tfg-natalia:~# _
```

Figura 85: Parámetros disponibles de errores en Squid

El que se utilizará en el proyecto será ERR_ACCESS_DENIED. En la siguiente figura vemos qué mensaje aparecerá por pantalla cada vez que nos bloquee una URL:

```
  GNU nano 4.6    /usr/share/squid/errors/es-es/ERR_ACCESS_DENIED     Modified

<html><head>
<meta type="copyright" content="Copyright (C) 1996-2020 The Squid Software Foun>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>ERROR: El URL solicitado no se ha podido conseguir</title>
<style type="text/css"><!--
 %l

body
:lang(fa) { direction: rtl; font-size: 100%; font-family: Tahoma, Roya, sans-se>
:lang(he) { direction: rtl; }
 --></style>
</head><body id="%c">
<div id="titles">
<h1>PRUEBA DE TFG</h1>
<h2>No se tiene permiso para acceder a esa pagina web</h2>
</div>
<hr>

<div id="content">
<p>Se encontró el siguiente error al intentar recuperar la dirección URL: <a hr>

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Figura 86: Mensaje de error al rechazar el acceso a la URL - Squid

ANEXO C: PUESTA EN FUNCIONAMIENTO DE LA HERRAMIENTA DE CONTROL PARENTAL

El objetivo de este anexo es indicar todos los pasos para poner en funcionamiento la herramienta de control parental que hemos ido desarrollando durante toda la memoria:

1. En primer lugar, **arrancamos el equipo antiguo conectando el USB** con la mini-distribución Alpine Linux. Si el equipo no estuviese configurado para arrancar mediante memoria USB, tendríamos que configurar la BIOS para que así fuese. Cuando termine de cargar la distribución, nos pedirá un usuario y contraseña. En este caso, los dos son "root".
2. La distribución ya cuenta en su interior con el directorio configurado (CarpetaTFG) que Docker necesita para funcionar. Esa carpeta es necesaria para que Docker pueda crear el volumen compartido entre su imagen y la máquina. Si listamos su contenido:
    - **paginasbloqueadas.txt:** Listado que el usuario irá rellenando mediante la página web para ir bloqueando las respectivas URLs.
    - **recargarsquid.txt:** Fichero que la aplicación utiliza como bandera para saber cuándo se ha añadido una nueva URL en paginasbloqueadas.txt. Cada vez que haya una modificación en el archivo, se tendrá que recargar Squid para que se apliquen los cambios. Para ello, se utiliza el siguiente script.
    - **script.sh:** Se trata de un script que comprueba el contenido de recargarsquid.txt. Si contiene la palabra "yes" significa que es necesario un reinicio del proxy y el script se encargará de hacer un "service squid reload". En la Figura 53 podemos ver con más detalle el contenido del script.
    - **url1 y url2:** Listados de un total de más de 10.000 URLs recomendadas por expertos que debe ser bloqueado su acceso. Squid ya se encuentra configurado para desbloquear directamente esas páginas webs.
3. Descargamos del repositorio de Docker, Docker Hub, la última versión de la aplicación. Para ello: >>**docker pull natsanpor/tfg.2:mytag**
4. Iniciamos squid:

    >>**service squid start**
5. Comprobamos la IP que tiene configurada en ese momento el servidor: ifconfig. Al estar configurada la IP por DHCP puede ir cambiando. Es necesario conocerla ya que con esa IP accederemos a la aplicación web.
6. Desplegamos la aplicación:

    >>**docker run --network="host" -v /root/CarpetaTFG/:/opt/CarpetaTFG -it natsanpor/tfg.2:mytag bash -p 4443:4443**
7. Si todo funciona correctamente, nos aparecerá un icono que pondrá "Spring Boot" y se quedará el proceso abierto.
8. Para acceder a la aplicación desde cualquier dispositivo conectado a la misma red que el servidor web:

    https://IP-Servidor:4443

    Así ya podremos disfrutar de nuestra herramienta de control parental.

    Para apagar la distribución: >> **poweroff**

REFERENCIAS

[1] SoftwareLab. ¿Qué Es Un Servidor Proxy Y Qué Significa? // Softwarelab. [online] Available at: https://softwarelab.org/es/servidor-proxy/

[2] ¿Qué es un servidor proxy y cómo funciona?. *¿Qué Es Un Servidor Proxy Y Cómo Funciona?*. [online] Available at: https://www.avast.com/es-es/c-what-is-a-proxy-server#topic-6

[4] Apuntes Fundamentos de Internet

[5] Apuntes Servicios Telemáticos Avanzados

[6] Qustodio. *La Mejor Herramienta De Control Parental Gratis De Internet*. [online] Available at: https://www.qustodio.com/es/

[7] Control parental Android para móvil y tablet | SecureKids. *Control Parental Android Para Móvil Y Tablet | Securekids*. [online] Available at: https://securekids.es/

[8] Eset.com. *Protege A Tus Hijos Online.*. [online] Available at: https://www.eset.com/es/hogar/parental-control-android/

[9] Ecured.cu. *Minidistribuciones Linux - Ecured*. [online] Available at: https://www.ecured.cu/Minidistribuciones_linux

[10] Puppylinux.com. *Puppy Linux Home*. [online] Available at: http://puppylinux.com/

[11] lubuntu. *Lubuntu*. [online] Available at: https://lubuntu.net/

[12] Bezencon, J., *Linux Lite Easy To Use Free Linux Operating System*. [online] Linuxliteos.com. Available at: https://www.linuxliteos.com/

[13] Peppermintos.com. *Peppermint OS – The Linux Desktop OS*. [online] Available at: https://peppermintos.com/

[14] Linux Adictos. 2020. *Las Mejores Distribuciones Linux Ligeras Y Rápidas, ¿Las Conoces Todas?*. [online] Available at: https://www.linuxadictos.com/distribuciones-ligeras.html

[15] ADSLZone. 2020. *Las Mejores Distribuciones De Linux Para Todo Tipo De Usuarios*. [online] Available at: https://www.adslzone.net/reportajes/software/mejores-distros-linux/

[16] Alpinelinux.org. 2020. *About | Alpine Linux*. [online] Available at: https://alpinelinux.org/about/

[17] Wiki.alpinelinux.org. 2020. *Alpine Linux*. [online] Available at: https://wiki.alpinelinux.org/

[18] Spring. 2020. *Spring Makes Java Simple.*. [online] Available at: https://spring.io/

[19] Docker. 2020. *Empowering App Development For Developers | Docker*. [online] Available at: https://www.docker.com/

[20] Javier Garzas. 2020. *Entendiendo Docker. Conceptos Básicos: Imágenes, Contenedores, Links… - Javier Garzas*. [online] Available at: https://www.javiergarzas.com/2015/07/entendiendo-docker.html

[21] Mejorsoftware.info. 2020. Alternativas A Squid . [en línea] Disponible en: https://mejorsoftware.info/tools/squid

[22] Es.slideshare.net. 2020. *Curso Squid Avanzado*. [online] Available at: https://es.slideshare.net/miguelangelnieto/curso-squid-avanzado.

[23] Squid-cache.org. 2020. *Squid : Optimising Web Delivery*. [online] Available at: http://www.squid-cache.org/

[24] Álvarez, C., 2020. *¿Qué Es Maven?*. [online] Genbeta.com. Available at: https://www.genbeta.com/desarrollo/que-es-maven

[25] Java desde 0. 2020. *Guía Básica De Maven - Java Desde 0*. [online] Available at: http://javadesde0.com/guia-basica-maven/

[26] Docs.spring.io. 2020. *18. Using The @Springbootapplication Annotation*. [online] Available at: https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-using-springbootapplication-annotation.html

[27] Docs.spring.io. 2020. *Springbootapplication (Spring Boot 2.3.4.RELEASE API)*. [online] Available at: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/SpringBootApplication.html

[28] Winch, R., 2020. *Hello Spring Security Java Config*. [online] Docs.spring.io. Available at: https://docs.spring.io/spring-security/site/docs/5.0.16.RELEASE/guides/html5/helloworld-javaconfig.html.

[29] Docs.spring.io. 2020. *Controller (Spring Framework 5.2.9.RELEASE API)*. [online] Available at: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Controller.html

[30] Docs.spring.io. 2020. *Getmapping (Spring Framework 5.2.9.RELEASE API)*. [online] Available at: https://docs.spring.io/spring-framework/docs/current/javadoc-pi/org/springframework/web/bind/annotation/GetMapping.html

[31] Docs.spring.io. 2020. *Postmapping (Spring Framework 5.2.9.RELEASE API)*. [online] Available at: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html

[32] Docs.spring.io. 2020. *Requestmapping (Spring Framework 5.2.9.RELEASE API)*. [online] Available at: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html

[33] RedesZone. 2020. *Conoce Esta Mega Recopilación De Hosts Para Navegar Más Tranquilos Bloqueando Webs*. [online] Available at: https://www.redeszone.net/2017/09/17/conoce-esta-mega-recopilacion-hosts-navegar-mas-tranquilos-bloqueando-webs/

[34] File.io. 2020. *File.Io - Ephemeral File Sharing*. [online] Available at: https://www.file.io/

[35] Alpinelinux.org. 2020. *Downloads | Alpine Linux*. [online] Available at: https://alpinelinux.org/downloads/

[36] Spring.io. 2020. *Spring Tools 4 Is The Next Generation Of Spring Tooling*. [online] Available at: https://spring.io/tools

[37] DigitalOcean. 2020. *Cómo Instalar Y Usar Docker En Ubuntu 18.04 | Digitalocean*. [online] Available at: https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-18-04-1-es.

[38] Wiki.alpinelinux.org. 2020. *Docker - Alpine Linux*. [online] Available at: https://wiki.alpinelinux.org/wiki/Docker