ULPGC
Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Aplicación web de auditoría de control de accesos

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Raúl González Espino

TUTORIZADO POR: Francisco Javier Carreras Riudavets

25 de noviembre de 2023

AGRADECIMIENTOS

A mi tutor, Francisco Javier Carreras Riudavets, pues su ayuda para entender mejor la tecnología y el servicio han sido vitales para el desarrollo y culminación del proyecto.

RESUMEN

Consiste en el desarrollo de una aplicación Web en la que se pueda controlar los accesos a varias páginas web. El sitio realizará esta tarea mediante la dirección IP del usuario y el nombre del dominio, estipulando un límite de consultas por IP. Si el límite de consultas es superado en una franja de tiempo que determinaremos previamente, se pasará la IP a la blacklist. También se implementará una función para poder bloquear y desbloquear manualmente una IP. El objetivo del TFT es la implementación de una web que funcione para facilitar las auditorías y el control de accesos de un servidor a los administradores de una web. Para dicha finalidad, debe cumplir con ciertos criterios como una interfaz amigable y funcionalidades como: un protocolo de lista blanca y negra para filtrar las posibles IPs malintencionadas, el bloqueo manual de estas y monitorización de las consultas de una o varias webs por su IP.

ABSTRACT

It consist on the developement of a web application in which access to various web pages can be controlled. The site will complete this task by taking the user IP and the name of the domain, setting a consultation limit for every IP. If the consultation limit is broken in a time interval that we would previously set, the IP will go to the blacklist. A function to block or unblock an IP manually will be implemented too. The main goal of this TFT is the implementation of a web that makes the auditories and the access control to a server easier for the web administrators to manage. For this purpouse it has to meet certain requirements like a friendly interface and functionalities like: a protocol of white list and black list to filter posible malicious IPs, the manual block of them and the monitorization of one or more webs by their IP direction.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El trabajo de fin de grado aquí presentado surge a modo de expansión del trabajo de fin de grado de otro alumno. El cuál sería: “Diseño e implementación de un Servicio web de prevención de ataque a un servidor” por Álvaro Santana Naranjo.

1.1 Origen de la idea

Este anterior trabajo tenía la intención de desarrollar un Servicio Web para controlar los accesos a diversas páginas didácticas de un servidor, las cuales poseen información valiosa. Debido al temor de que esta información acabe siendo robada y usada para replicar el trabajo se optó por realizar un algoritmo que detectara factores como: el número y método de accesos de los usuarios y el tipo de conexión para estableciendo ciertos límites negar el acceso a aquellos que se los salten. Un protocolo de lista blanca o lista negra.

El algoritmo fue desarrollado con éxito, también se creó una primera versión de una página web en la que solo se implementó una rudimentaria función de monitorización. Es decir, solo se mostraban las tablas de la base de datos.

Este proyecto trata de mejorar esa web añadiendo funcionalidades nuevas que le sean de utilidad al administrador tales como poder modificar la base de datos, buscar registros concretos y bloquear direcciones IP sospechosas, además de realizar la conexión con el servicio.

1.2 Objetivos iniciales

El objetivo principal del TFT era la implementación de una web que facilite las auditorías y el control de accesos de un servidor a los administradores de una web, para dicha finalidad se plantearon los siguientes criterios como:

- Una interfaz amigable:

Lo cual se ha cumplido pues la interfaz diseñada es muy descriptiva y sencilla de entender.

-Funcionalidades como: un protocolo de lista blanca y negra para filtrar posibles IPs malintencionadas, el bloqueo manual de estas y monitorización de las consultas de una web por IP.

Todas estas funcionalidades fueron implementadas correctamente, y además se implementaron nuevas como: control de la base de datos (botones de insertar, editar y eliminar), filtrado de campos concretos y generación de informes en pdf.

2. ESTADO ACTUAL

En la situación previa al desarrollo del servicio web nos encontrábamos con un servidor en el que se alojan varias aplicaciones didácticas, las cuales han sido víctimas de numerosos ataques informáticos. Un ejemplo de estos servicios sería la página TuLengua.es que aloja 6 herramientas didácticas.

Figura 2.1: Aplicaciones didácticas de TuLengua.es

Estas aplicaciones tienen una enorme cantidad de información que llevó mucho tiempo ser recolectada, por lo que son víctimas de ataques realizados por arañas. Las arañas son programas que tienen la capacidad de inspeccionar páginas web y recopilar la información que en ellas se almacena. Aunque tampoco es la única manera de la que son atacadas.

Antes de implementar el servicio todo lo que se hacía era bloquear las direcciones ip sospechosas mirando los logs. Lo cual puede ser bastante ineficiente a la par que tedioso, de ahí surgió la idea de realizar un algoritmo que realizara esta gestión de forma automática.

Con estos problemas en mente el anterior alumno desarrolló un servicio web en el que se incluye dicho algoritmo y que funciona reenviando los accesos detectados a diversas webs del servidor a un archivo de texto.

Todo ocurría de forma automática mientras el servicio estuviera activo y se dieran los accesos. Se podía monitorizar de formas poco amigables para el usuario (pruebas impresas por consola, abriendo el archivo de texto, etc).

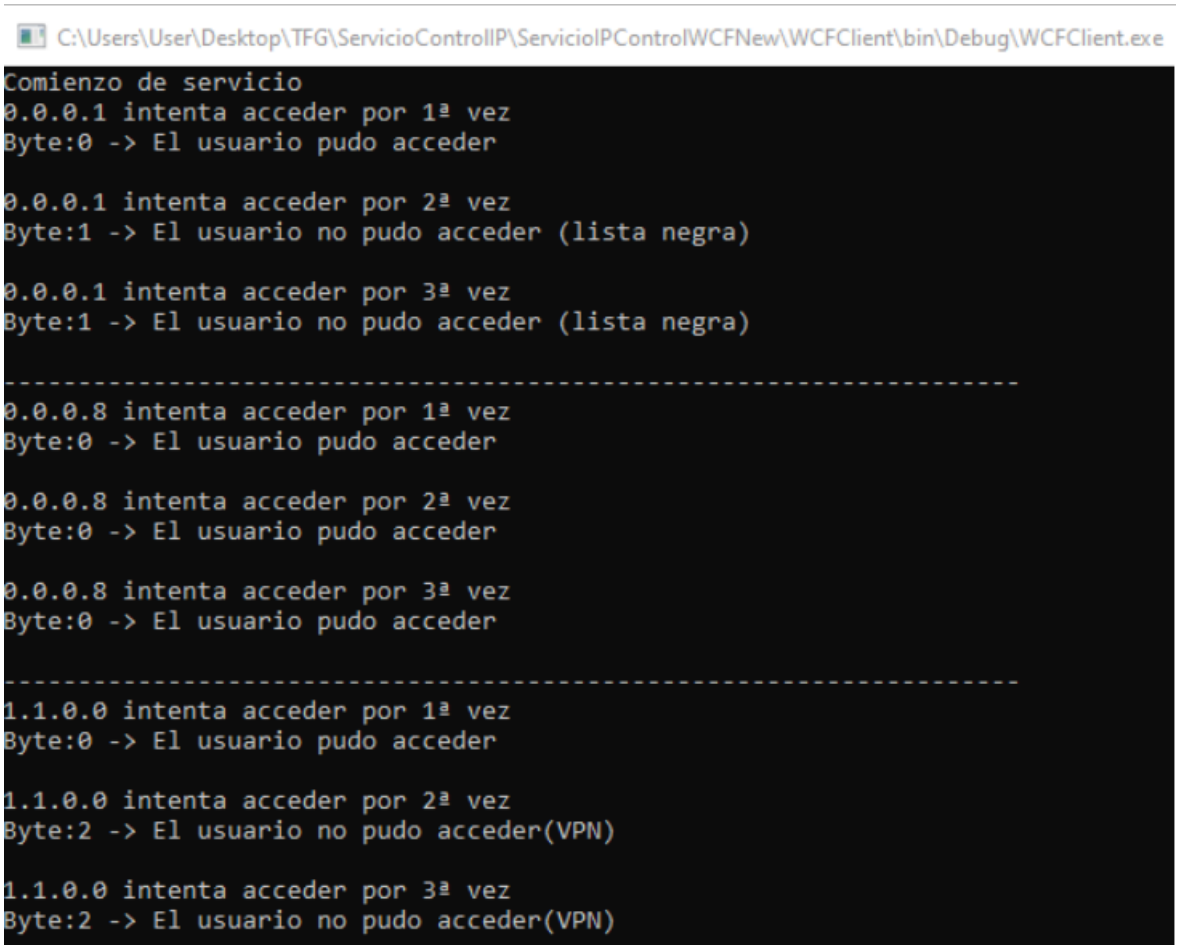


Figura 2.2: Últimas pruebas de ejecución del servicio

Además, cuando empezó el desarrollo de la web se creó una interfaz robusta para mostrar cada lista en un apartado, incluso se hizo distinción de castigos permanentes y temporales en la lista negra mediante un método getPunishment(); el cual detectando la url en la que está el usuario en ese momento devuelve un string que con un if se le dicta al formulario web con que datos rellenar las tablas en el método PageLoad().

Figura 2.2: Muestra de los tipos de castigo en Blacklist

Actualmente no se utiliza un código para rellenar la tabla en el PageLoad, pues se ha optado por el uso de GridViews que llenan una tabla de datos automáticamente mediante el uso de SqlDataSource (fuentes de datos Sql es el nombre del componente, pero nosotros utilizaremos Access como herramienta gestora de base de datos).

Los GridView facilitan bastante el trabajo porque, además de mostrar los datos sin necesidad de programar su inserción en una tabla, tienen entre otras facilidades la posibilidad de generar botones para ejecutar instrucciones como: eliminar, editar, seleccionar e insertar.

3. COMPETENCIAS

A continuación, explicaré como se han visto reflejadas las competencias especificadas en el TFT01:

·**CII01** (Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente). Se refleja en el proyecto en que se desarrolla una aplicación web, se diseñan diversas funcionalidades y se realizan las pruebas necesarias para comprobar su fiabilidad hasta que el resultado sea el esperado. Comprobando además que no causen interferencias en la ejecución de otras funciones.

·**CII08** (Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados). El lenguaje de programación y las tecnologías ya estaban condicionados por el trabajo previo del compañero. Por otra parte, la robustez, eficiencia y seguridad ha sido muy importante en el desarrollo, pues se ha tenido que hacer muchas pruebas para poder imaginar qué clase de errores podrían darse cuando un usuario navegue por la página y evitar que se den interrupciones.

·**CII17**(Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas). Esta competencia podría decirse que se ha cumplido por el diseño de una interfaz sencilla y descriptiva para que los usuarios tengan claro su funcionamiento con solo verla, y la configuración de alertas para evitar que se realicen acciones que no corresponden con el flujo de ejecución esperado.

·**TI07**(Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos). La finalidad misma del trabajo es crear una página que monitoriza una base de datos de la que a su vez dependen los protocolos de lista blanca y negra que controlan los accesos abusivos a las páginas de un servidor, bloqueando el acceso a aquellas direcciones IP que se sobrepasen de los límites impuestos, ergo garantizando así su seguridad.

4. APORTACIONES DEL TRABAJO

El aporte de este Trabajo de Fin de Grado radicaría en convertir un servicio web que ya funcionaba por sí mismo controlando los accesos del servidor y convertirlo en una web en la que el administrador tenga un entorno mucho más amigable con el que trabajar.

Ha evolucionado desde una aplicación en la que se guardaban los accesos en un archivo de texto y se añadía una ip a la blacklist cuando se pasaban de consultas a ser una web en la que: muestra cada tabla de la base de datos en su propia sección; se pueden eliminar, editar o insertar nuevos registros manualmente; bloquear direcciones si se considera necesario e incluso generar informes en pdf.

Podríamos resumir los aportes de este trabajo entonces en: un mayor control en la monitorización sobre lo que ocurre en el servicio y una mucho más mayor capacidad de acción y reacción ante posibles incidencias en el servicio dándole así al administrador un rol mucho más activo en la defensa del servidor.

5. DESARROLLO

La página, como ya se ha dicho anteriormente, tiene la principal finalidad de mostrar las tablas de la base de datos y además se le han añadido varias funcionalidades: para la modificación de sus registros (eliminar, editar o insertar), bloquear direcciones IP, generar informes en pdf y filtrar por ciertos campos.

En este punto de describirán aspectos relacionados con el desarrollo como: las tecnologías y metodologías utilizadas, fases de desarrollo, explicaciones de puntos clave del código.

5.1 Tecnologías y metodologías

Las tecnologías usadas en este proyecto del desarrollo de la web serían las siguientes: .NET Framework versión 4.7.2 en el front-end, el lenguaje de programación C# para el back-end, Access 2016 para la gestión de la base de datos y el entorno de desarrollo el Visual Studio 2022.

Quitando el hecho de que estas tecnologías estaban condicionadas por el trabajo anterior del desarrollo del servicio web, es fácil entender por qué se eligió la combinación de Asp.Net y Visual Studio pues este entorno concede múltiples ayudas y atajos para el susodicho Framework. Framework el cuál, se podría usar con varios lenguajes distintos en el backend como Python, Visual Basic o Ruby, pero se optó por C# por mayor familiaridad.

“.NET Framework es un entorno de ejecución runtime que administra aplicaciones cuyo destino es .NET Framework. Incorpora Common Language Runtime, que proporciona la administración de la memoria y otros servicios del sistema, y una biblioteca de clases completa, que permite a los programadores aprovechar el código estable y fiable de todas las áreas principales del desarrollo de aplicaciones.” [1]

Sobre Access poco hay que decir, siendo que .Net Framework es un marco de trabajo desarrollado por Microsoft y Visual Studio el entorno de desarrollo propio de Microsoft una herramienta también perteneciente a esta compañía era el complemento perfecto.

Además, para el método de generación de pdf cabe resaltar que también he utilizado la librería iText7, que se usa para tratar con documentos digitales tanto en C# como en Java.

"iText para Java representa el siguiente nivel de SDK para desarrolladores que quieran aprovechar los beneficios que puede aportar el PDF. Equipado con un mejor motor de documentos, capacidades de programación de alto y bajo nivel y la capacidad de crear, editar y mejorar documentos PDF, iText puede ser de gran ayuda para casi todos los flujos de trabajo." [2]

5.2 Fases del desarrollo

En el planteamiento del proyecto se indicaron las fases que seguiría, y se han cumplido en su totalidad. Estas fases han son:

1.**Estudio previo y análisis**: no había usado nunca antes ASP.NET ni Access, así que el primer paso fue aprender lo necesario viendo tutoriales y haciendo programas de menor escala hasta aprender como configurar un GridView para mostrar la base de datos. Duró entre 2 o 3 semanas, luego de ello me reuní con el tutor para que me indicara como proceder.

2. **Diseño, desarrollo e implementación**: esta fase ha sido la más larga debido a muchos factores: En alguna ocasión hubo que replantearse algún que otro método una vez programado debido a que pensábamos que se podía pulir más. También en la forma de implementar las funciones de la página a veces había algún fallo o bien interferencia con otras funcionalidades que impedían que funcionara correctamente, tuve que recurrir a la ayuda del tutor varias veces para encontrar los motivos y parchearlos. Por otra parte, cuando por fin conectamos el servicio web tuvimos un pequeño inconveniente, ya que hubo que cambiar nombres de campos de alguna tabla de la base de datos y por ende sus referencias en el código de algunas funcionalidades.
Todo lo anterior en conjunto hizo que esta fase durara desde principios de agosto hasta finales de octubre.

3. **Evaluación, Validación y Pruebas**: La fase de pruebas se ha compaginado con el desarrollo, cada vez que se implementaba una función se probaba hasta que el resultado fuera el esperado. Además, cuando descubrí la primera interferencia entre funciones empecé a comprobar que en las ejecuciones cuando se modificara una funcionalidad que no afectase a las demás. No fue hasta obtener una versión con las funciones acordadas implementadas y sin interferencia que el tutor me dio por válido el código para escribir la memoria.

4. **Documentación y Presentación**: en esta última fase escribo la memoria y preparo la presentación. Tenía claro desde el principio algunos de los temas que tenía que mencionar, como lo de que este trabajo es una continuación de otro. Otros han ido surgiendo conforme el desarrollo se ha dado y los he ido apuntando.

La metodología elegida ha sido una ágil, ya que no trabajábamos ni con plazos fijos, más que reuniones semanales o cada 2 semanas. Tampoco teníamos objetivos cerrados a cambios, cosa que como he mencionado en puntos anteriores hizo que implementara la generación de documentos pdf.

Y dentro de las metodologías ágiles la elegida fue SCRUM, pero una versión más básica y edulcorada ya que solo dividimos el desarrollo en sprints orientativamente, ya que como mencioné anteriormente se dieron circunstancias que me hicieron tener que volver a retocar algunas funciones. Aquí una tabla que mostrará los sprints:

SPRINT	META	DURACIÓN (Semanas aprox.)
1er	Configuración de los Gridview con las bases de datos	1
2º	Añadir botones de ordenar, eliminar, editar e insertar	1
3º	Filtrar	4
4º	Bloquear	3
5º	Generación de docs pdf	2
6º	Paginación	3
7º	Comunicación con el servicio	2

Tabla 5.1: Aproximación de los sprints

Los *sprints* se daban por terminados cuando se daba por valido el funcionamiento de su función protagonista. Luego en reuniones con el tutor conceptualizábamos los objetivos del siguiente sprint, a veces también se nos ocurrían mejoras y otras veces eran reuniones para terminar de arreglar problemas. Para aclaraciones menos complejas se utilizaron el correo o llamadas de teams.

5.3 Desarrollo de la web

Como se ha mencionado en puntos anteriores la estructura de la web estaba planificada y parcialmente creada antes de que yo tomara las riendas del proyecto.

En este punto se profundizará en todo lo relacionado al desarrollo de la web: Las funcionalidades que se han desarrollado, el flujo de ejecución de cada uno, los componentes en el front-end necesarios para su funcionamiento, las librerías externas utilizadas en el caso que se de dicho caso, diferencias entre versiones de las funciones para distintos formularios, problemas con los que nos encontramos durante su implementación, etc.

Pero antes de llegar a ello es necesario explicar con detalle la estructura de la versión previa de la web y los cambios realizados antes de que se empezase a añadir las nuevas funciones.

5.3.1 Estructura de la web

La web tiene la estructura “formulario web con página maestra” lo que implica que está compuesta por una página básica (Site.Master) que renderiza el contenido de varios formularios que muestran cada uno una tabla distinta: Whitelist, VPNlist, Blacklist (dividida en castigos permanentes y temporales) y Logs. En cada formulario hay una tabla que muestra los datos de la base de datos. Y en la parte superior a estas hay varias cajas de texto para escribir los datos por los que filtrar además de más botones para realizar acciones.

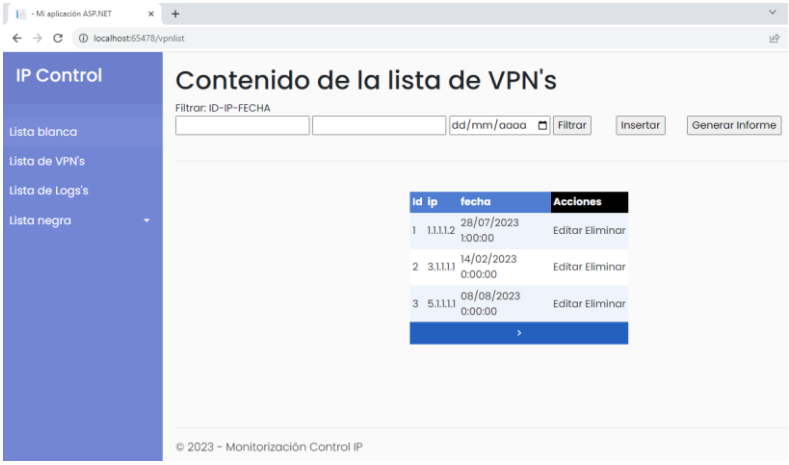


Figura 5.1: Ejemplo del estado actual de la web

El principal cambio con la versión anterior de la web: El formulario de Logs se ha añadido en esta nueva versión, para lo cual tuve que modificar la página maestra para añadir su enlace al menú principal de la página.

```
<ul class="list-unstyled components">
    <li>
        <a href="/whitelist.aspx">Lista blanca</a>
    </li>
    <li>
        <a href="/vpnlist.aspx">Lista de VPN's</a>
    </li>
    <li>
        <a href="/Logs.aspx">Lista de Logs's</a>
    </li>
    <li>
        <a href="#pageSubmenu" data-toggle="collapse" aria-expanded="false" class="dropdown-toggle">Lista negra</a>
        <ul class="collapse list-unstyled" id="pageSubmenu">
            <li>
                <a href="/blacklist.aspx?castigo=temporal">Direcciones castigadas</a>
            </li>
            <li>
                <a href="/blacklist.aspx?castigo=permanente">Lista negra</a>
            </li>
        </ul>
    </li>
</ul>
</nav>
```

Figura 5.2: Modificación realizada en el Site.Master

Por otra parte, las barras de búsqueda y los botones con acciones no estaban antes y en el desarrollo de ellos es donde más ha radicado el trabajo realizado en este caso.

Figura 5.3: Ejemplo del estado previo de la web

Por otro lado, la mayor diferencia (que también se mencionó anteriormente) es que en la versión previa se rellena una tabla mediante un código en el método PageLoad (en este marco de trabajo es el método de backend que se ejecuta por defecto cuando la página se carga), mientras que en esta nueva versión es la combinación de un componente llamado GridView con otro llamado SqlDataSource. Del código backend del compañero no he usado nada excepto el método getPunishment que tiene la función de devolver si estamos en la parte de castigos temporales o en la de permanentes.

```
4 referencias
protected string getPunishment()
{
    string url = HttpContext.Current.Request.RawUrl;
    int index = url.IndexOf("=");
    if (index < 0)
    {
        return "permanente";
    }
    else
    {
        return url.Substring(index + 1);
    }
}
```

Figura 5.4: Método getPunishment()

Este método es de vital importancia porque nos ahorra hacer 2 formularios distintos para temporal y permanente. En su lugar solo se necesitará condicionante “If” para que se ejecute una versión distinta de cada funcionalidad para cada url.

De igual manera, cabe resaltar que en mi versión de la base de datos he añadido a parte de los campos que venían dados por la versión anterior un campo Id, el cual es un número que sirve para marcar el orden en el que fueron añadidos los registros en la tabla.

En VPN y whitelist solamente se registran la Id, la IP y la fecha en la que se añadió, mientras que en la blacklist y en los logs aparte de esos hay un par de campos más:

-**permanente**: numero entero entre 0 y 1. 0 implica un castigo temporal para la IP y 1 significará un castigo permanente.

-**accesos**: es el número total de accesos de la ip al servidor.

-**accesos_fallidos**: es el número total de accesos fallidos al servidor de la ip.

- **accesos_boton:** es el número de accesos totales al servidor mediante el uso de botones.

- accesos_url: es el número de accesos totales al servidor mediante el uso de enlaces url.

- ultimo_acceso: es la fecha y hora del último acceso de la ip a una página del servidor.

- **periodo:** periodo medio de la frecuencia de accesos al servidor.

- **ids_sesion:** cantidad de ids de sesión con los que accede la ip al servidor.

-**control:** string de código de control.

5.3.2 GridView y sus instrucciones

A continuación, se explicará más a detalle el código frontend y todo sobre el componente GridView a detalle. Partamos de su definición: “El GridView control se usa para mostrar los valores de un origen de datos en una tabla. Cada columna representa un campo, mientras que cada fila representa un registro.” [3]

Ese origen de datos puede ser un archivo xml o una base de datos como en este caso. A continuación, en la siguiente figura se mostrará un GridView configurado.

Figura 5.5: Configuración de un GridView

Como se puede apreciar en la figura he marcado de amarillo el DataSourceID, pues es el componente por el que se le insertarían datos a la tabla. En nuestro caso ese componente es el SqlDataSource1, el cual es un componente del mismo nombre destinado a conectar fuentes de datos con aquellos otros que enlacen con datos como GridView o DetailsView (el cual se explicará más adelante). En nuestro caso la configuración será así:

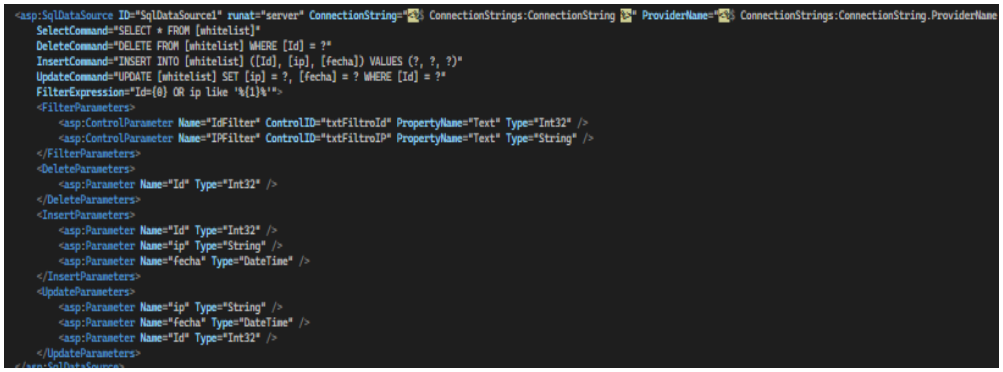


Figura 5.6: Configuración de un SqlDataSource

En la figura se puede observar que se accede a la base de datos mediante el parámetro connectionStrings, que está situado en el Web.Config y el ProviderName es el nombre del proveedor de datos, en nuestro caso System.Data.OleDb. También se aprecia que hay varias instrucciones para ejecutar en la base de datos, son aquellas que se ejecutan en los botones generados por el GridView. En nuestro caso las acciones implementadas gracias a GridView son “Eliminar” y “Editar” en todos los formularios, en WhiteList además tenemos el botón de “Seleccionar” pero se explicará en detalle un par de puntos más adelante.

La opción del botón de insertar también estaba disponible, pero al no terminar de convencer la estética de la funcionalidad se optó por implementarla de otra forma:

Para empezar, creamos un formulario nuevo llamado “Insertar_NombreDeLaTabla” para cada formulario ya existente, exceptuando el de Logs debido a que su tabla está destinada a llenarse sola con los accesos de IPs que se vayan dando (en su lugar tendrá un botón “Actualizar”).

Estos nuevos formularios solo tendrán 3 componentes: un botón “Volver”, un SqlDataSource y un componente llamado DetailsView que al igual que GridView sirve para enlazar datos en bases de datos.

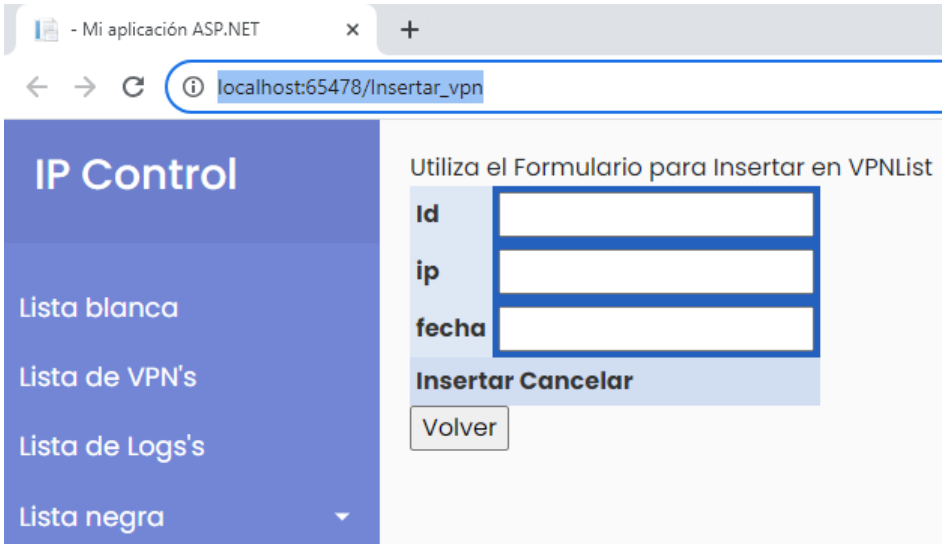


Figura 5.7: Muestra del Formulario Insertar_VPNList

Para más contexto con respecto al componente DetailsView tenemos su definición en esta referencia de Docs Microsoft:

“Muestra los valores de un único registro de un origen de datos en una tabla, donde cada fila de datos representa un campo del registro. El control DetailsView permite editar, eliminar e insertar registros.” [4]

Tiene varios modos, pero nosotros nos vamos a centrar en el modo “Insert” pues es la finalidad que buscamos. El DataSource será uno con la configuración igual que el del formulario de la página homónima.

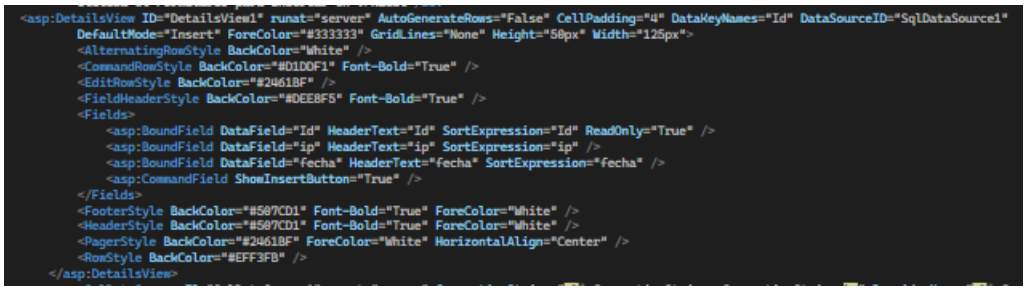


Figura 5.8: Configuración de un DetailsView

Luego de tener el formulario de inserción correctamente configurado simplemente es necesario añadir un botón “Insertar” en el formulario de la tabla, cuya única función sea llevar al usuario al formulario de inserción de datos. Una vez dentro podrán usar el DetailsView para añadir un registro. Para volver al formulario original solo será necesario usar el botón “Volver”.

Las configuraciones de GridView y DetailsView mostradas como ejemplo han sido las de VPNList, la configuración es igual en WhiteList, pero en BlackList y Logs hay diferencias:

Para empezar, Logs no tiene la posibilidad de insertar manualmente así que no necesita configurar el DetailsView. Lo siguiente es recordar que teniendo campos distintos en las tablas ambos componentes estarán configurados para esos campos en cada respectivo caso.

Y para terminar es importante explicar que Blacklist necesita de 2 parejas de GridView y SqlDataSource, una para el castigo temporal y otra para el castigo permanente.Para ello se hace uso del método getPunishment con un condicional “If” si se está en permanente se mostrará el GridView las filas con el campo “permanente” a 1 y en temporal las que tengan este campo a 0.

Para que se seleccionen solo las filas que deseamos en cada tabla, en los DataSource simplemente se deberá tener configurado un parámetro de selección con el valor que se busca utilizar en cada caso, y en la instrucción de Select indicar que se mostrarán en la tabla solo aquellas filas cuyo campo “permanente” tenga el valor del parámetro.

```
SelectCommand="SELECT * FROM [blacklist] WHERE ([permanente] = ?)"
 <SelectParameters>
     <asp:Parameter DefaultValue="0" Name="permanente" Type="Int32" />
```

Figura 5.9: Expresión de Selección y Parámetro de Selección

(Nota: Hay más parámetros de selección configurados en los DataSource de Blacklist, pero serán explicados en el método de filtro).

5.3.3 Función de filtrado

Para implementar este método se utilizan los siguientes componentes: 3 textBox en caso de Whitelist y VPNList (ID, IP y Fecha) y 4 en el caso de Logs y Blacklist (los mismos 3 anteriores + ids_sesion).

```
<asp:TextBox ID="txtFiltroId" runat="server" TextMode="Number" ></asp:TextBox>
<asp:TextBox ID="txtFiltroIP" runat="server" ></asp:TextBox>
<asp:TextBox ID="txtFiltroFecha" runat="server" TextMode="Date" ></asp:TextBox>
<asp:TextBox ID="txtFiltroIds" runat="server" TextMode="Number" Width="127px" ></asp:TextBox>
```

Figura 5.10: Configuración de TextBox en diferentes modos

Es importante especificar los modos de texto correspondiente para cada valor para así controlar que el usuario no pueda introducir valores inesperados que causen una interrupción. Además, se deben de explicar ciertos componentes del SqlDataSource: Los parámetros de selección, filtro y expresión de filtro.

```
SelectCommand="SELECT * FROM [vpnlist]"
DeleteCommand="DELETE FROM [vpnlist] WHERE [Id] = ?"
InsertCommand="INSERT INTO [vpnlist] ([Id], [ip], [fecha]) VALUES (?, ?, ?)"
UpdateCommand="UPDATE [vpnlist] SET [ip] = ?, [fecha] = ? WHERE [Id] = ?"
FilterExpression=" Id={0} OR ip like '%{1}%'">
<FilterParameters>
    <asp:ControlParameter Name="IdFilter" ControlID="txtFiltroId" PropertyName="Text" Type="Int32" />
    <asp:ControlParameter Name="IPFilter" ControlID="txtFiltroIP" PropertyName="Text" />
</FilterParameters>
```

Figura 5.11: Expresión y Parámetros de filtro

Con los parámetros de filtro podemos indicarle al programa que ciertos componentes, en nuestro caso los textBox, introducirán datos en una expresión de filtrado la cual se ocupará de filtrar la tabla por esos valores.

En la figura tenemos la expresión definida por defecto para poder tener los filtros de los parámetros Id e IP establecidos. La cual significa que el valor numérico de Id debe ser exacto al que se inserte en el filtro, mientras que IP solo debe contener el string que se inserte, da igual si al inicio, al final o dentro.

La expresión se verá sometida a cambios en el método en el que aplicamos el filtro de búsqueda, pues hay más campos por los que filtrar.

A continuación, explicaré parte por parte el algoritmo diseñado para ello:

```
// Obtener los valores de los TextBoxes
string filtroId = txtFiltroId.Text.Trim();
string filtroIP = txtFiltroIP.Text.Trim();
string filtroFecha = txtFiltroFecha.Text.Trim();

if (filtroId.Length == 0 && filtroIP.Length == 0 && filtroFecha.Length == 0)
{
    SqlDataSource1.FilterParameters["IdFilter"].DefaultValue = "";
    SqlDataSource1.FilterParameters["IPFilter"].DefaultValue = "";
    SqlDataSource1.FilterExpression = "Id={0} OR ip like '%{1}%'";
    return;
}
else
{
```

Figura 5.12: Método de filtrado de campos Parte 1

Para empezar, obtenemos el texto que haya escrito en los textBoxs, y hacemos una comprobación en la que si resulta que todos los textBoxs estuvieran vacíos dejaremos la expresión de filtro como estaba por defecto y los parámetros de filtro a un valor de string vacío. Esto tiene la finalidad de que cuando hayamos usado cualquier filtro podamos devolver la tabla a su estado inicial con simplemente borrar lo que escribimos en los textBox y hacer click en el botón de nuevo. En caso contrario, que al menos uno de los textbox no esté vacío, se ejecutará el código del “else”:

```
    int idFilter = 0;
    if (int.TryParse(filtroId, out idFilter))
    {
        SqlDataSource1.FilterParameters["IdFilter"].DefaultValue = idFilter.ToString();
    }
    else
    {
        SqlDataSource1.FilterParameters["IdFilter"].DefaultValue = "0";
    }
    if (!string.IsNullOrEmpty(filtroIP))
    {
        SqlDataSource1.FilterParameters["IPFilter"].DefaultValue = filtroIP;
        // Si el filtro de IP está vacío, configurar el filtro para que coincida con cualquier valor de IP
    }
    else
    {
        SqlDataSource1.FilterParameters["IPFilter"].DefaultValue = "a";
    }

    // Convertir el filtro de Fecha a DateTime
    DateTime fechaFilter;
    string fechaFilterIni, fechaFilterEnd;

    if (DateTime.TryParse(filtroFecha, out fechaFilter))
    {
        //SqlDataSource1.FilterParameters["DateFilter"].DefaultValue = fechaFilter.ToString();
        fechaFilterIni = fechaFilter.ToString("MM-dd-yyyy");
        fechaFilterEnd = fechaFilter.AddDays(1).ToString("MM-dd-yyyy");
    }
    else
    {
        fechaFilterIni = "01/01/0001 0:00:00";
        fechaFilterEnd = "01/01/0001 0:00:00";
        //SqlDataSource1.FilterParameters["DateFilter"].DefaultValue = "01/01/0001 0:00:00";
    }


    // Aplicar el filtro solo si al menos uno de los campos de filtro no está vacío o es diferente de su valor predeterminado
    SqlDataSource1.FilterExpression = "Id = {0} OR ip like '%{1}%' OR (fecha >= #" + fechaFilterIni + "# AND fecha < #" + fechaFilterEnd + "#)";

    // Refrescar los datos del GridView
    GridView1.DataBind();
}
```

Figura 5.13: Método de filtrado de campos Parte 2

Luego de hacer las comprobaciones de que el texto escrito pueda ser convertido al tipo de valor que es cada campo, si es válido se añadirá el valor al parámetro de filtro y en caso contrario se añadirá un valor que es imposible que se dé para que no se tome en cuenta. Esto en el caso de los campos que hayan sido definidos en los parámetros de filtro, es decir todos menos las fechas. Las fechas tienen el problema de que al ser del tipo "DateTime", es decir vienen en conjunto una fecha y una hora del día, así que para evitarle al usuario tener que escribir una hora concreta se opta por hacer un intervalo entre la hora que se escriba en el textBox y el dia siguiente, para que así se muestre todos los registros de ese día independientemente de su hora. Así que luego de realizar todas estas comprobaciones solo quedaría actualizar la expresión de filtro para que tenga en cuenta el intervalo de las fechas y actualizar los datos del GridView.

Y la explicación de este método podría terminar aquí pero el de Blacklist y Logs es algo distinto:
Para empezar, tienen un campo de más por el que filtrar, el cuál es ids_sesion, que es un entero así que el funcionamiento sería como el de Id en ese aspecto.

```
FilterExpression=" Id={0} OR ip like '%{1}%' OR ids_sesion={2}">
<FilterParameters>
    <asp:ControlParameter Name="IdFilter" ControlID="txtFiltroId" PropertyName="Text" Type="Int32" />
    <asp:ControlParameter Name="IPFilter" ControlID="txtFiltroIP" PropertyName="Text" />
    <asp:ControlParameter Name="IdsFilter" ControlID="txtFiltroIds" PropertyName="Text" Type="Int32"/>
</FilterParameters>
```

Figura 5.14: Expresión y Parámetros de Filtro de Logs

Aunque cabe destacar que Blacklist no utiliza parámetros de filtro sino de selección, esto debido a que como en ambos DataSource creamos el parámetro de selección, para que se muestren solo castigos permanentes o temporales, y alteramos la expresión de selección no se detectaba la expresión de filtro al añadirla, así que se optó en este caso por simplemente usar la expresión de selección.

```
SelectCommand="SELECT * FROM [blacklist] WHERE [permanente] = ? AND( Id >= ? OR ip LIKE ? OR ids_sesion >= ?)"
<SelectParameters>
    <asp:Parameter DefaultValue="1" Name="permanente" Type="Int32" />
    <asp:ControlParameter Name="IdFilter" ControlID="txtFiltroId" PropertyName="Text" Type="Int32" DefaultValue="0" />
    <asp:ControlParameter Name="IPFilter" ControlID="txtFiltroIP" PropertyName="Text" Type="String" DefaultValue="%"/>
    <asp:ControlParameter Name="IdsFilter" ControlID="txtFiltroIds" PropertyName="Text" Type="Int32" DefaultValue="0"/>
```

Figura 5.15: Expresión y Parámetros de Seleccion de Blacklist

A continuación, mostraré los cambios en esta adaptación del método para Blacklist. Para evitar ser redundante se omitirá la comprobación de los campos que vimos anteriormente ya que su único cambio es que donde antes ponía “FilterParameter” ahora pone “SelectParameters”:

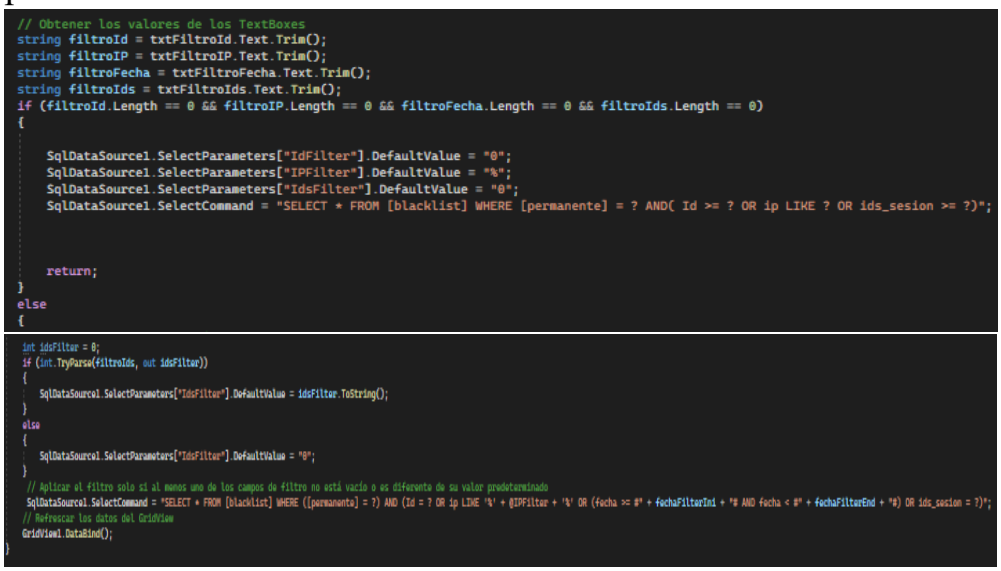


Figura 5.16: Cambios en Blacklist del método de filtrado

La lógica de la expresión de selección es algo distinta a la de filtro, pues en este caso además de que se cumpla uno de los criterios de búsqueda que necesitamos tenemos que tener en cuenta que el valor del permanente siempre sea el mismo. Además, el símbolo “?” no estaba funcionando para seleccionar la IP, porque obtiene el string sin los símbolos “%” ergo no se detecta correctamente, así que se usa el nombre del parámetro @IPFilter y los símbolos “%” para obtener el resultado esperado.
Además de todo esto cabe resaltar que se usó de nuevo el método getPunishment para distinguir entre las 2 versiones del método:

Figura 5.17: Evento onClick del botón filtrar de Blacklist

Las únicas diferencias entre ambos métodos son los DataSource y los GridViews donde se aplican los cambios.

5.3.4 Generación de documentos pdf

Esta función fue creada con la idea de generar registros periódicos del estado de las tablas consultables en todo momento. Como explicamos anteriormente en las tecnologías utilizadas usamos métodos de la librería iText7, que se utiliza para generar y trabajar con documentos digitales. A continuación, procederé a explicar el código de este método:

```
protected void btnPdf_Click(object sender, EventArgs e)
{
    DataView dv;
    if (getPunishment() == "permanente")
    {
        dv = (DataView)SqlDataSource1.Select(DataSourceSelectArguments.Empty);
    }
    else
    {
        dv = (DataView)SqlDataSource2.Select(DataSourceSelectArguments.Empty);
    }
    DataTable dt = dv.ToTable();

    // Crear un MemoryStream para almacenar el PDF
    using (MemoryStream ms = new MemoryStream())
    {
        // Crear un nuevo documento PDF
        PdfDocument pdfDoc = new PdfDocument(new PdfWriter(ms));
        Document doc = new Document(pdfDoc);

        // Agregar titulo al informe
        if(getPunishment() == "permanente")
        {
            doc.Add(new Paragraph("Estado de Blacklist: Castigos permanentes"));
        }
        else
        {
            doc.Add(new Paragraph("Estado de Blacklist: Castigos temporales"));
        }


        // Crear una tabla para representar los datos
        iText.Layout.Element.Table table = new iText.Layout.Element.Table(dt.Columns.Count, false);

        // Configurar el ancho de la tabla para ocupar toda la página
        table.SetWidth(UnitValue.CreatePercentValue(100));
```

Figura 5.18: Método generar informe Parte 1

En esta primera parte del código se puede apreciar de nuevo el getPunishment, pues usaremos de referencia la variante del método en Blacklist. En las otras versiones las asignaciones que se hagan aquí dependiendo de este método se harán sin más pues los demás formularios solo tienen un DataSource.

Volviendo a la explicación, el primer paso del método es crear usando el SqlDataSource un DataView, para a su vez con este último crear una tabla. Luego de esto se creará un MemoryStream que es una estructura que nos ayuda para almacenar datos en memoria, en nuestro caso el Pdf.
Creamos el documento pdf y procedemos a agregarle el título, luego de esto crearemos una tabla en la que se representarán los datos y configuramos el ancho de la tabla para que ocupe todo el del documento.

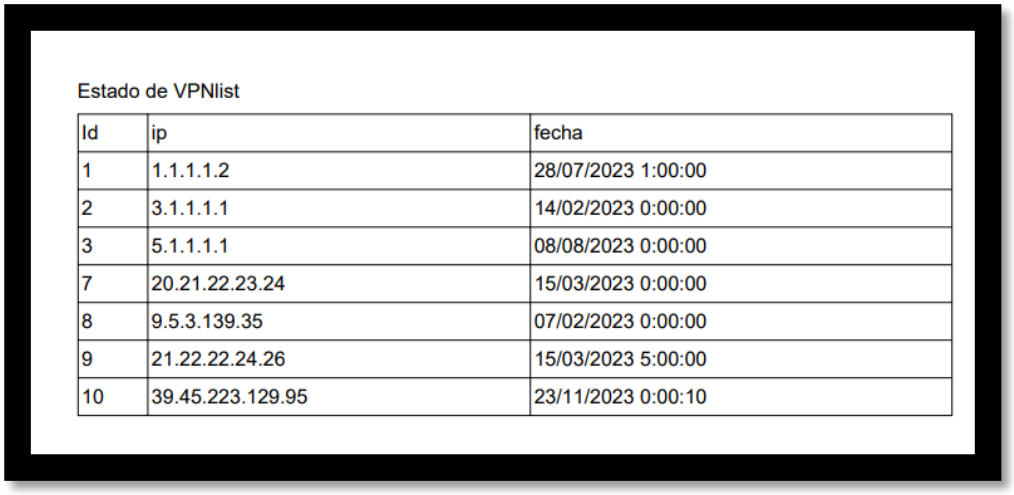
```
    // Agregar encabezados de columna
    //Se reduce el tamaño de fuente para que toda la tabla quepa
    foreach (DataColumn col in dt.Columns)
    {
        if (col.ColumnName.Length > 8)
        {
            table.AddCell(new Cell().Add(new Paragraph(col.ColumnName).SetFontSize(5)));
        }
        else
        {
            table.AddCell(new Cell().Add(new Paragraph(col.ColumnName).SetFontSize(7)));
        }
    }

    // Agregar los datos al PDF en forma de tabla
    foreach (DataRow row in dt.Rows)
    {
        foreach (DataColumn col in dt.Columns)
        {
            table.AddCell(new Cell().Add(new Paragraph(row[col].ToString())));
        }
    }

    // Agregar la tabla al documento
    doc.Add(table);

    // Cierra el documento
    doc.Close();

    // Envía el PDF al navegador para descarga
    Response.ContentType = "application/pdf";
    if (getPunishment() == "permanente")
    {
        Response.AddHeader("Content-Disposition", "attachment; filename=informe_blacklist_Castigos_permanentes " + DateTime.Now.ToString() + ".pdf");
    }
    else
    {
        Response.AddHeader("Content-Disposition", "attachment; filename=informe_blacklist_Castigos_temporales " + DateTime.Now.ToString() + ".pdf");
    }
    // Response.AddHeader("Content-Disposition", "attachment; filename=informe_blacklist "+DateTime.Now.ToString()+".pdf");
    Response.BinaryWrite(ms.ToArray());
    Response.Flush();
    Response.End();
}
```

Figura 5.19: Método generar informe Parte 2

Lo siguiente en el proceso sería añadir los nombres de las columnas en las tablas, en la figura se puede apreciar que se cambia el tamaño de la fuente dependiendo del tamaño de las palabras. Esto fue una solución de último minuto a un problema que hubo con el renombramiento de campos en la base de datos, ya que los nombres nuevos eran muy grandes y hacían que la tabla no se viera bien, apareciendo cortada. Es algo para lo que habría que investigar mejoras para futuras actualizaciones, por suerte en VPNList o Whitelist no tiene ese problema porque solo tiene 3 campos.

Luego de esto se añaden los datos a la tabla, se agrega la tabla al documento, se cierra y se envía al navegador para su descarga luego de nombrarlo. Al nombrarlo añadimos la fecha y hora de creación del informe para que así se pueda llevar un control sobre las versiones de la tabla.

Aquí un ejemplo de cómo quedaría un informe:

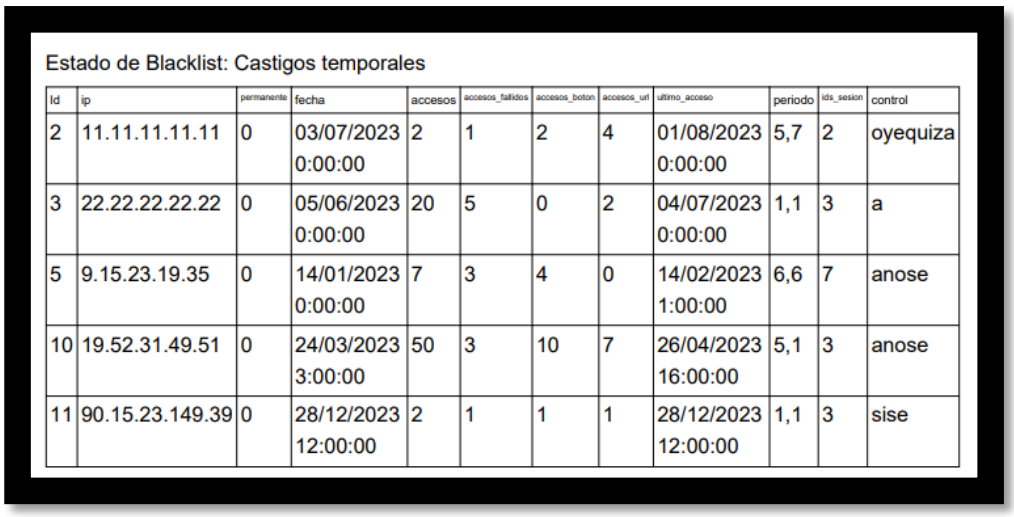


Estado de VPNlist

Id	ip	fecha
1	1.1.1.1.2	28/07/2023 1:00:00
2	3.1.1.1.1	14/02/2023 0:00:00
3	5.1.1.1.1	08/08/2023 0:00:00
7	20.21.22.23.24	15/03/2023 0:00:00
8	9.5.3.139.35	07/02/2023 0:00:00
9	21.22.22.24.26	15/03/2023 5:00:00
10	39.45.223.129.95	23/11/2023 0:00:10

Figura 5.20: Informes generados en VPNList por el método

Aquí un ejemplo de cómo quedaría el informe de blacklist:



Estado de Blacklist: Castigos temporales

Id	ip	permanente	fecha	accesos	accesos_fallidos	accesos_boton	accesos_url	ultimo_acceso	periodo	ids_sesion	control
2	11.11.11.11.11	0	03/07/2023 0:00:00	2	1	2	4	01/08/2023 0:00:00	5,7	2	oyequiza
3	22.22.22.22.22	0	05/06/2023 0:00:00	20	5	0	2	04/07/2023 0:00:00	1,1	3	a
5	9.15.23.19.35	0	14/01/2023 0:00:00	7	3	4	0	14/02/2023 1:00:00	6,6	7	anose
10	19.52.31.49.51	0	24/03/2023 3:00:00	50	3	10	7	26/04/2023 16:00:00	5,1	3	anose
11	90.15.23.149.39	0	28/12/2023 12:00:00	2	1	1	1	28/12/2023 12:00:00	1,1	3	sise

Figura 5.21: Informes generados en Blacklist por el método

Lo de encoger los nombres es una solución temporal, se explorarán posibles soluciones en el apartado destinado a posibles expansiones futuras.

5.3.5 Bloqueo IP y modificación del servicio

La finalidad de este método es seleccionar una fila de una de las tablas de registros sin castigos como la Whitelist o los Logs para eliminarla y añadirla a Blacklist en caso de que el administrador sospeche de la actividad de esa IP. Requiere varios componentes para que funcione así que se explicará parte por parte.

Para empezar los componentes necesarios en el front end de la página son el botón de seleccionar y el botón de bloquear.

Figura 5.22: Vistazo a botones de Seleccionar y BloquearIp

El botón Bloquear IP bloqueará la fila seleccionada mediante el método que programamos en su evento OnClick, pero antes de explicar un par de cosas antes.

Primero, el botón de seleccionar es generado por el GridView y tiene el defecto de que normalmente solo deselecciona una fila cuando se selecciona una fila nueva, esto se ha solucionado con el siguiente código:

```
0 referencias
protected void GridView1_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
{
    if (GridView1.SelectedIndex == e.NewSelectedIndex)
    {
        GridView1.SelectedIndex = -1;
        e.Cancel = true;

    }
}
```

Figura 5.23: Método de Deselección

Con este evento hacemos que si el índice seleccionado nuevo es igual al que estaba previamente seleccionado lo cambiaremos a -1 para que no coincida con ninguno de la tabla. Además, debemos poniendo e.Cancel a true para desactivar cualquier evento que no esté bajo nuestro control.

Para continuar explicaré la modificación realizada al servicio web del compañero:

```
namespace ServicioIPControlWCFNew
{
    [ServiceContract]
    1 referencia
    public interface IServicioIPControl
    {
        [OperationContract]
        1 referencia
        byte TryAccess(Access value);

        [OperationContract]
        1 referencia
        int BlockIp(int id, string ip, DateTime date, int permanent, int blocked);

    }
```

Figura 5.24: Contratos de operación en IServicioIPControl.cs

En este archivo se declara la infraestructura del servicio y debemos de declarar los métodos principales del servicio que se vayan a ejecutar fuera de este como contrato de operación. TryAccess es el método principal, su función es comprobar que una dirección Ip tenga permisos de acceso. Y BlockIp es el método que hemos añadido nosotros, es necesario declararlo aquí para que podamos acceder a él más tarde, ahora se procederá a explicar su contenido:

```
1 referencia
public int BlockIp(int id, string ip, DateTime date, int permanent, int blocked)
{
    InfoHash info = new InfoHash();
    if (whiteList.Contains(ip))
    {
        whiteList.Remove(ip);
        blackList.Add(ip);
        return 1;
    }
    else
    {
        return 0;
    }

}
```

Figura 5.25: Método de blockIp añadido a ServicioIPControl.svc.cs

Un poco de contexto: El servicio para sus gestiones automáticas almacena las direcciones IP de cada tabla en un HashSet. Con este método lo único que hacemos es eliminar el registro temporal de WhiteList (en caso de que contenga la IP en su lista) para añadirlo en Blacklist. Para que el bloqueo se complete todavía haría falta modificar la base de datos, para ello habría que volver al código backend de la página:

```
0 referencias
protected void btnBloquear_Click(object sender, EventArgs e)
{
    GridViewRow selectedRow = GridView1.SelectedRow;

    if (selectedRow != null)
    {
        // Obtener los valores de las celdas en la fila seleccionada
        string id = selectedRow.Cells[0].Text; // Id está en la primera columna
        string ip = selectedRow.Cells[1].Text; // IP está en la segunda columna
        string fecha = selectedRow.Cells[2].Text; // Fecha está en la tercera columna


        // Configurar conexion a la base de datos
        string connectionString = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
        //Conexion al servicio web
        int blocked= 0;
        blocked=ServicioIPControl.BlockIp( int.Parse(id), ip,DateTime.Parse(fecha), 1, blocked);
        if (blocked == 0)
        {
            ScriptManager.RegisterStartupScript(this, this.GetType(), "alert", "alert('No se bloqueo la ip');", true);
            return;
        }
        else
        {
```

Figura 5.26: Método para bloquear Ip en la página Parte 1

Obtenemos la fila seleccionada del GridView y separamos las celdas en variables. Obtenemos la cadena de conexión y conectamos con el servicio para eliminar los registros temporales, en caso de que no sea posible el método enviará una alerta y se detendrá.

```
// Consultas SQL
string deleteQuery = "DELETE FROM [whitelist] WHERE [Id] = ?";
string insertQuery = "INSERT INTO [blacklist] ([Id], [ip], [permanente], [fecha], [accesos], [accesos_fallidos], [accesos_boton], [accesos_url], [ul

try
{
    using (OleDbConnection connection = new OleDbConnection(connectionString))
    {
        //Abrir conexion
        connection.Open();

        // Primera consulta: DELETE
        using (OleDbCommand deleteCommand = new OleDbCommand(deleteQuery, connection))
        {
            // Parámetro para la consulta DELETE
            deleteCommand.Parameters.AddWithValue("@Id", id);

            // Ejecutar la consulta DELETE
            int rowsAffected = deleteCommand.ExecuteNonQuery();

        }

        // Segunda consulta: INSERT
        using (OleDbCommand insertCommand = new OleDbCommand(insertQuery, connection))
        {
            // Parámetros para la consulta INSERT
            insertCommand.Parameters.AddWithValue("@Id", id);
            insertCommand.Parameters.AddWithValue("@IP", ip);
            insertCommand.Parameters.AddWithValue("@permanent", 1);
            insertCommand.Parameters.AddWithValue("@Fecha", fecha);
            insertCommand.Parameters.AddWithValue("@access", 0);
            insertCommand.Parameters.AddWithValue("@fail", 0);
            insertCommand.Parameters.AddWithValue("@button", 0);
            insertCommand.Parameters.AddWithValue("@url", 0);
            insertCommand.Parameters.AddWithValue("@last", fecha);
            insertCommand.Parameters.AddWithValue("@periodos", 0.0);
            insertCommand.Parameters.AddWithValue("@ids", 1);
            insertCommand.Parameters.AddWithValue("@control", "bloqueado");

            // Ejecutar la consulta INSERT
            int rowsInserted = insertCommand.ExecuteNonQuery();

        }
        // Refrescar los datos del GridView
        GridView1.DataBind();
        //Cerrar conexion
        connection.Close();
    }
```

Figura 5.27: Método para bloquear Ip en la página Parte 2

Una vez hecho el ajuste de registros en el servicio procederemos a hacerlo en la base de datos. En la figura se aprecia el proceso, parece complicado debido a la gran cantidad de líneas de código, pero es bastante sencillo en realidad:

Lo primero de todo sería declarar las query en dos variables de string. Para continuar, abriremos la conexión a la base de datos haciendo uso de la estructura "using". Al igual que le hacemos un "using(OleDbConnection)" para conectar a la base de datos, ahora debemos hacer un "using(OleDbCommand)" para usar los comandos.

El primero en ejecutarse será "delete", para evitar que haya dos direcciones IP iguales a la vez en whitelist y blacklist. Se le añade el parámetro de la Id a la query y se ejecuta.

En el caso de "insert" el proceso es el mismo, la única diferencia serían los parámetros. La mayoría son numéricos y al desconocer el total de accesos se ha puesto a 0 todo menos: los ids, porque mínimo ha de tener uno; y el permanente, debido a que si se ha elegido bloquearlo el administrador debe tener sus razones. Sobre los otros parámetros tenemos: fecha y ultimo_acceso que ambos tendrán el mismo valor. Y además tenemos el código de control que simplemente dirá: bloqueado.

Luego de ejecutar la query todo lo que quedaría es actualizar los datos del GridView con un DataBind y cerrar la conexión.

Cabe resaltar que en caso de que han configurado alertas en caso de que no se pudiera conectar con la base de datos o no hubiera ninguna fila seleccionada.

```
            }
            catch (Exception ex)
            {
                // Manejo de excepciones
                // Mostrar una alerta
                ScriptManager.RegisterStartupScript(this, this.GetType(), "alert", "alert('Ha ocurrido una excepción.');", true);
            }

        }
    }
    else
    {
        // No se ha seleccionado ninguna fila en el GridView
        //Mostrar una alerta
        ScriptManager.RegisterStartupScript(this, this.GetType(), "alert", "alert('Seleccione una fila.');", true);
    }
```

Figura 5.28: Método para bloquear Ip en la página Parte 3

5.3.6 Paginación y Ordenar

Estas dos funcionalidades son independientes la una de la otra, pero como no son muy extensas de explicar compartirán el apartado. Aunque cierto es que tienen algo en común, ambas son configuradas desde el propio GridView.

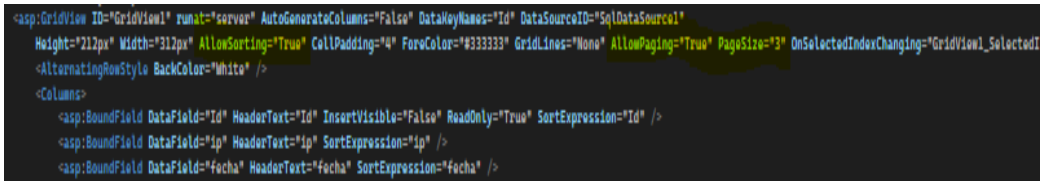


Figura 5.29: Parámetros de paginación en el GridView

Como se puede apreciar basta con simplemente cambiar las variables “AllowSorting” y “AllowPaging”. En el caso de la paginación también se definió el tamaño de página a tres filas, esto debido a que la base de datos no era muy extensa y se necesitaba hacer pruebas con varias páginas.

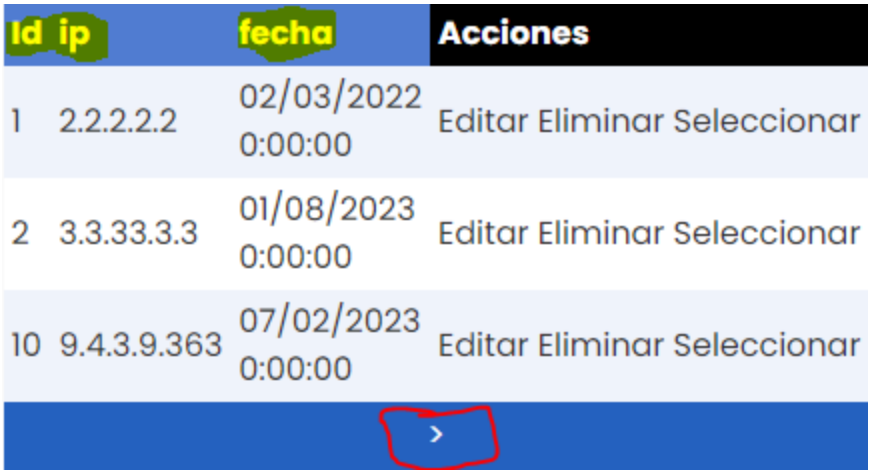


Figura 5.30: Botones de ordenar y pasar página

En la figura está la tabla con unas marcas, en amarillo tenemos los nombres de los campos, haciendo click en ellos se ordenará la tabla en función ascendente o descendente por ese campo. De ahí a que el campo Id sea tan importante.

Por otra parte, rodeado en rojo tenemos la flecha que pasa las páginas, solo va a la inmediatamente siguiente o a la anterior (en la imagen no aparece la anterior porque es la primera página).

6. POLÍTICAS DE ACTUACIÓN

En este punto se explicarán las razones por las que se tomaron varias decisiones durante el desarrollo de la web. Algunas de ellas se tomaron por simple estética, otras tantas sin embargo definieron la forma en la que el proyecto iba a funcionar, ya sea por cómo se conceptualizaron o como se acabaron programando.

6.1 Inspiración en herramientas SIEM

Antes de ahondar en este punto habría que definir qué consideramos una herramienta SIEM:

"SIEM (información de seguridad y gestión de eventos), es una tecnología capaz de detectar rápidamente, responder y neutralizar las amenazas informáticas. Su objetivo principal es el de proporcionar una visión global de la seguridad de la tecnología de la información." [5]

Básicamente sería un tipo de herramienta de seguridad que permite al administrador tener un rol activo en la seguridad de sus servidores, tal y como esta web. Aunque siendo sinceros tampoco se ha tenido muy presente en el desarrollo, solamente en la funcionalidad de generar informes en pdf. Esto debido a que durante las prácticas de empresa trabajé con una de estas herramientas y como tenía dicha función pensé que sería buena idea contar con ella, para que el administrador pueda poseer un registro con fechas marcadas de los cambios que sufran las tablas de la base de datos.

6.2 Insertar con DetailsView

La decisión de realizar la funcionalidad de insertar registros nuevos en las tablas viene debido a 2 razones, la primera por obvias razones es para evitar tener que hacer un formulario nuevo desde cero configurando los textbox, botones, conexión a la base de datos, etc. El DetailsView junto con el SqlDataSource lo hace todo por nosotros.

Todo eso suena muy bien, pero la gran pregunta sería: ¿por qué no usar el propio GridView para ello? Pues como ya se dijo anteriormente fue una decisión puramente estética, ya que con el botón de insertar la tabla se vería más o menos así:

ID	IP	FECHA	ACCIONES
1	1.1.1.1.1	4/10/2023 13:15:00	Eliminar Editar Nuevo
2	2.2.2.2.2	24/1/2023 0:00:00	Eliminar Editar Nuevo
3	3.3.3.3.3	16/11/2023 0:00:00	Eliminar Editar Nuevo

Tabla 6.1: Ejemplo de tabla GridView con el botón insertar autogenerado

No se llegó a probar que funcionara, pero abría un menú similar al de Editar. En caso de que funcionase como editar no nos valdría para nada, y en caso de que funcionara sigue siendo extraño tener tantos botones de insertar cuando todos cumplen la misma función. Por ello se optó por el DetailsView como una opción más viable.

También es importante comentar que se decidió meter al DetailsView en otro formulario también por un tema estético ya que la página se veía muy sobrecargada con ambos componentes View juntos.

6.3 Filtrado por campos

Esta es sencilla de explicar, al principio el filtro estaba configurado, pero solo se había planteado para el campo de la dirección Ip. Luego de una reunión con el tutor subimos la apuesta para filtrar también por ID y Fecha. Y luego de esto, cuando empezamos a programar la versión de blacklist se optó también por configurar el filtro del número de ids de sesión de cada Ip.

Cabe resaltar también la cantidad de variantes que hay de este método, al principio se tenía planeado solamente hacerlo con los FilterParameters pero como empezaron a dar error en la versión de Blacklist opté por hacer una versión con parámetros de selección para no quedarme atascado en busca de la causa del error y así agilizar un poco el proceso.

También es digno de mención que en estas funciones de filtrado hubo algún momento en el que se interfería con otras funciones: Al volver a filtrar con los textos vacíos hubo algún momento en el que no se devolvían todos los datos del GridView, lo que causaba interferencias como que los informes se generaban en blanco y solo se cargaba la primera página del GridView. Por suerte se pudo arreglar. Aquí un ejemplo de cómo se veía el informe cuando ocurría este error:

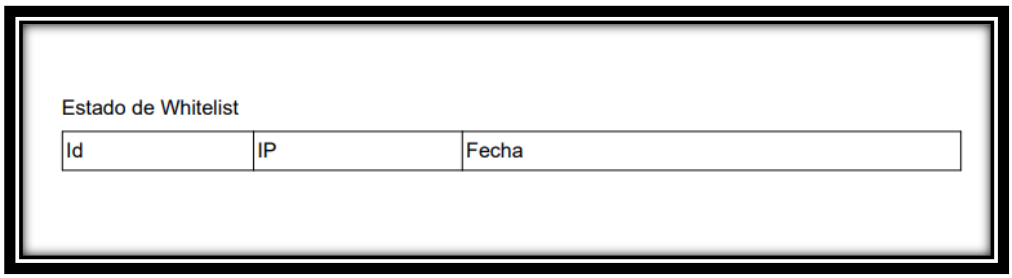
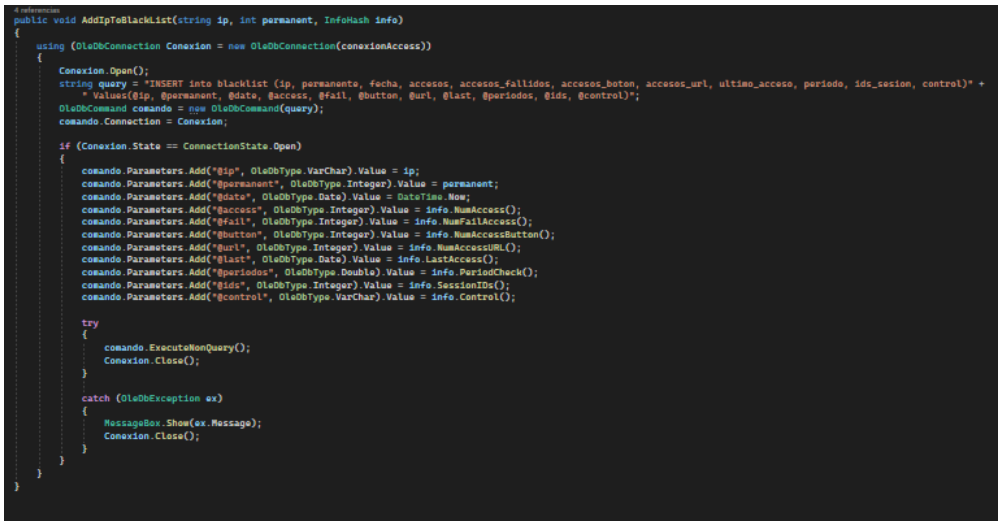


Figura 6.2: Informe generado después de un error con el filtrado

6.4 Refactorización de campos en la base de datos

Durante la creación de la base de datos del proyecto tomé como referencia este método del servicio web para crear los campos:



```
4 referencias
public void AddIpToBlackList(string ip, int permanent, InfoHash info)
{
    using (OleDbConnection Conexion = new OleDbConnection(conexionAccess))
    {
        Conexion.Open();
        string query = "INSERT into blacklist (ip, permanente, fecha, accesos, accesos_fallidos, accesos_boton, accesos_url, ultimo_acceso, periodo, ids_sesion, control)" +
            " Values(@ip, @permanent, @date, @access, @fail, @button, @url, @last, @periodos, @ids, @control)";
        OleDbCommand comando = new OleDbCommand(query);
        comando.Connection = Conexion;

        if (Conexion.State == ConnectionState.Open)
        {
            comando.Parameters.Add("@ip", OleDbType.VarChar).Value = ip;
            comando.Parameters.Add("@permanent", OleDbType.Integer).Value = permanent;
            comando.Parameters.Add("@date", OleDbType.Date).Value = DateTime.Now;
            comando.Parameters.Add("@access", OleDbType.Integer).Value = info.NumAccess();
            comando.Parameters.Add("@fail", OleDbType.Integer).Value = info.NumFailAccess();
            comando.Parameters.Add("@button", OleDbType.Integer).Value = info.NumAccessButton();
            comando.Parameters.Add("@url", OleDbType.Integer).Value = info.NumAccessURL();
            comando.Parameters.Add("@last", OleDbType.Date).Value = info.LastAccess();
            comando.Parameters.Add("@periodos", OleDbType.Double).Value = info.PeriodCheck();
            comando.Parameters.Add("@ids", OleDbType.Integer).Value = info.SessionIDs();
            comando.Parameters.Add("@control", OleDbType.VarChar).Value = info.Control();

            try
            {
                comando.ExecuteNonQuery();
                Conexion.Close();
            }

            catch (OleDbException ex)
            {
                MessageBox.Show(ex.Message);
                Conexion.Close();
            }
        }
    }
}
```

Figura 6.3: Método AddIpToBlackList del servicio web

Utilicé los nombres de los valores en vez de los de los campos debido a un error por mi inexperiencia con la tecnología y trabajé en base a ellos hasta que una vez fue necesario conectar con el servicio puse los correctos.

No quise modificar los nombres en el servicio debido a que era la vía más rápida hacerlo de este modo, pero mi opinión cambió luego de tener que reconfigurar el método de generar informe de blacklist. Expandiré más todo esto en la conclusión.

7. CONCLUSIÓN Y POSIBLES EXPANSIONES

Para concluir este informe primero quiero recalcar que estoy muy satisfecho con el trabajo realizado, pues en unos pocos meses he pasado de considerar el desarrollo web como una de mis más grandes debilidades en esta disciplina que conocemos como la ingeniería informática a ser capaz de añadir funciones nuevas a una web cuyo desarrollo comenzó de la mano de otra persona. Sí que es cierto que estaba buscando un trabajo enfocado más en la ciberseguridad, y aunque varios conceptos han estado presentes durante el planteamiento y la implementación es innegable que la parte más predominante ha sido el desarrollo web.

También cabe resaltar de nuevo la ayuda brindada por el tutor, ya que en momentos clave sirvió para esclarecer errores de los que no era capaz de ver la solución por mucho que investigase. Además de proporcionarme los materiales necesarios para conocer a fondo el contexto de todo el trabajo realizado por el anterior alumno en desarrollo del servicio web.

Por otra parte, es la primera vez que trabajo con las tecnologías que se han utilizado en el desarrollo de este proyecto: tanto como con el .Net Framework como con las bases de datos de Access y en cuanto a C#, aunque sí que había algo de experiencia previa no era ni de lejos mi lenguaje predilecto.

En el punto en el que estoy ahora con estas herramientas, terminando un trabajo de final de grado el cual empecé apenas habiendo oído hablar de estas tecnologías, puedo decir que no solo me he vuelto más hábil con estas, sino que además espero poder volver a trabajar con ellas.

Pero como cabe esperar, a pesar de que ahora me sienta así no significa que todo me resultara fácil durante el desarrollo del proyecto. Aprender a usar las herramientas me llevó algo de tiempo y también se empezó a programar sin una definición clara de cómo serían algunos métodos. Así que hubo que hacer una labor de ponerse en la piel del usuario para ver cómo se abordaba el desarrollo de algunas funciones.

También era necesario ponerse en el lugar del usuario para probar la interfaz a conciencia y predecir las formas no ideales de usar la página para así evitar fallos de funcionamiento o en el peor de los casos interrupciones. En concreto los fallos que encontré más frustrantes fueron aquellos que eran causados por interferencias entre métodos, es decir lo que ya se comentó que ocurrió con el filtrado que afectaba a la paginación y los informes.

A parte de ponerse en el lugar del usuario, ahora mientras escribo la memoria y le doy vueltas en mi mente al código se me ocurren algunas expansiones para mejorar la experiencia del usuario, o en este caso administrador de la página, o algunas expansiones que simplemente nos ahorrarían líneas de código. Aparte claro está de las correcciones que se deben hacer.

En cuanto a trabajo futuro: como prioridad estaría arreglar el generar informe de BlackList, pues el tener que cambiar los nombres de los campos por unos del doble de longitud era algo con lo que no contaba y por temas de tiempo no se encontró una mejor opción para mantener la tabla completa en el informe que no involucrase el cambiar los nombres de los campos del servicio.

También en el código de filtrado en blacklist creo que se debería cambiar la forma en la que se gestiona el tema de las dos variantes del código, pues es posible que sea mejor opción tener solamente un método de filtrado y que se pasen por parámetro tanto el GridView como el SqlDataSource con los que se trabaje en cada caso.

En otro orden de cosas se me han ocurrido varias ideas para las que ya sí que tendría que ponerme de acuerdo con el tutor para ver cuáles serían viables y cuáles no.

Por ejemplo, en el método de filtro tenemos que se filtra por id, ip, fecha y ids_sesion. Una idea sería modificar el filtro de id porque el tener que saberse el número exacto de la id de un registro es fácil cuando hay números pequeños, pero a las páginas del servidor que queremos proteger se puede acceder en cualquier momento así que me imagino que la cantidad de registros incrementará inevitablemente. La propuesta de mejora sería cambiar el tener que adivinar el número exacto de la id que queremos buscar por quizá un intervalo entre dos números, los números menores al introducido o algo similar. También planteo lo mismo para ids_sesion, aunque no creo que la cantidad de Ids distintas que pueda tener una sola Ip siempre es más amigable para el usuario poder ampliar la búsqueda a varias posibilidades y no simplemente introducir el número exacto.

Otra idea involucrando el método de filtro también sería filtrar por el campo de "control", el cual es un string por el cual se registra la razón del bloqueo (Bloqueado, Máximo de accesos permitidos, Accesos por tiempo, etc.). El tener este campo y no usarlo para filtrar es posiblemente desaprovecharlo y quizás convenga añadir dicha posibilidad.

Y ya para finalizar quedaría el formulario de Logs: faltaría por configurar el método de bloqueo de Ip para este formulario. La parte de eliminar en una tabla de la base de datos e insertar en otra no sería complicado configurarla, pero tendría que hablar con el tutor sobre como modificaríamos el servicio para este porque los accesos al servidor o logs se gestionan de forma diferente a las otras listas (WhiteList, BlackList y VpnList).

```
//Hash donde se van a almacenar los accesos al servidor
private static ConcurrentDictionary<string, InfoHash> ipHash = new ConcurrentDictionary<string, InfoHash>();
//Hash donde se van a almacenar el principio de las direcciones que accedan al servidor, para gestionar la multiIP
//La clave son los 3 primeros números de la dirección IP, y el Hashset las ips que acceden y empiezan por la clave
private static ConcurrentDictionary<string, HashSet<String>> multiIPHash = new ConcurrentDictionary<string, HashSet<String>>();
//Hashsets donde se van a guardar las listas de direcciones
public static HashSet<String> whiteList;    //Listas de direcciones permitidas
public static HashSet<String> blackList;    //Listas de direcciones que tiene prohibido el acceso al servidor
public static HashSet<String> vpnList;      //Lista de direcciones que son VPNs, que tampoco tienen acceso
```

Figura 7.1: Muestra de las tablas Hash del servicio web

También es importante decir que en el formulario de Logs se supone que se insertarán los logs que se vayan dando en el servidor pero que no se han configurado todavía para redirigirlos a dicha tabla de la base de datos. Consideraría este de las expansiones más cruciales de este trabajo.

En conclusión, la página a pesar de ese pequeño inconveniente del tamaño de las fuentes en los informes de BlackList está funcionando perfectamente en las funcionalidades que se han planteado en los objetivos iniciales del TFT01. Lo cual no significa que no se pueda estar abiertos a mejoras y nuevas ideas, pero eso dependerá de si le parecen correctas las sugerencias al futuro administrador de esta página.

8. BIBLIOGRAFÍA

[1] Genevieve, W., Coulter, D et al. (6 de abril de 2022). Introducción a .NET Framework. Recuperado de: Docs Microsoft https://learn.microsoft.com/es-es/dotnet/framework/get-started/ [11/11/2023]

[2] Radchuck, D., Subach,A. et al. (16 de marzo de 2021). About iText7. Recuperado de: GitHub https://github.com/itext/itext7 [12/11/2023]

[3] Anderson,R.(16 de mayo de 2023). Gridview Clase.Recuperado de: Docs Microsoft https://learn.microsoft.com/es-es/dotnet/api/system.web.ui.webcontrols.gridview?view=netframework-4.8.1 [15/11/2023]

[4] Anderson,R.(16 de mayo de 2023). Detailsview Clase. Recuperado de: Docs Microsoft https://learn.microsoft.com/es-es/dotnet/api/system.web.ui.webcontrols.detailsview?view=netframework-4.8.1 [17/11/2023]

[5] Polanco,C. (7 de abril del 2020).SIEM, la tecnología capaz de detectar y neutralizar las amenazas informáticas antes de que ocurran. Recuperado de: SOFECOM https://sofecom.com/que-es-un-siem/ [23/11/2023]