Capítulo 5:

PRUEBAS.

5.1 Objetivos de las pruebas.

Objetivos de las pruebas.

Hoy en día el tema de la seguridad en software ya no resulta nada nuevo, en los inicios los desarrolladores de software no procuraban dar seguridad a sus sistemas, ya que el riesgo de que el sistema fuera violado resultaba mínimo. Hoy en día, con la expansión de las redes y su utilización para la creación de sistemas ha provocado que las violaciones a los mismos sean más frecuentes haciendo necesaria la aplicación de medidas de seguridad.

La aplicación de medidas de seguridad no solo trata de cerrar el sistema contra posibles violaciones, sino también ayudan a evitar posibles fallos después de su implementación. Para lograr mejores resultados durante las pruebas del SIGEB (Sistema de Gestión de Becarios) se propondrán una serie de objetivos a cumplir al final de las pruebas, los cuales se enuncian a continuación:

- Depurar el sistema para disminuir en lo posible la aparición de errores.
- Procurar que la información que se maneja en el sistema se encuentre íntegra.
- Lograr un sistema que sea confidencial y seguro.

5.2 Diseño de casos de prueba.

Para poder realizar un diseño de pruebas más adecuado recordaremos qué hace y cómo funciona el Sistema de Gestión de Becarios (SIGEB).

Tareas del SIGEB:

- Realizar la evaluación de becarios de UNICA.
- Mostrar los reportes de actividades realizados por cada becario.
- Dar seguimiento a los proyectos que se realizan en UNICA.

Para realizar lo anterior se utilizarán las siguientes tecnologías:

- Lenguajes y scripts de programación: Java, JavaScript, AJAX, CSS y HTML.
- Base de datos: PostgreSQL.
- Tomcat.
- Para la creación de documentos: iReports.

¿Cómo funciona el SIGEB?
El Sistema De Gestión de Becarios (SIGEB) es un Sistema WEB, que para su implementación se necesitará de un software que pueda resolver los archivos JSP como HTML para mostrarlos en cualquier navegador WEB. Se basa en una arquitectura del tipo cliente-servidor, utilizando una base de datos relacional para el almacenamiento de los datos.

Lo primero que se debe hacer para poder utilizar el SIGEB es autenticarse en él. En esta parte de la autenticación se utiliza codificación JSP, CSS y HTML además de una conexión a una Bases de Datos. Una vez autentificado el usuario se le asigna una sesión, la cual le permitirá al sistema saber cuáles son sus privilegios de usuario dentro del sistema y las actividades que puede realizar.

La codificación utilizada una vez autentificado el usuario son los lenguajes JSP, AJAX, HTML y CSS. Los datos que se presentan y se manipulan se almacenan en una base de datos, la cual se encuentra en otro servidor, así que será necesaria la conexión entre ambos servidores, en el que se encuentra la aplicación y en el que se encuentra la base de datos.
Conociendo lo anterior, se podrá comenzar a formular los diseños de casos de las pruebas en base a las posibles vulnerabilidades que puede tener el SIGEB de acuerdo a las tecnologías que utiliza.

El primer caso de prueba que se aplicará es el de introducir datos aleatorios en los formularios del SIGEB, principalmente en el de acceso, que en cuestión de seguridad resulta el más vulnerable. Este caso de prueba buscará observar el comportamiento del SIGEB ante el procesamiento de datos no esperados. Para lo anterior se aplicará la técnica de Caja Negra, llamando de la misma forma a este caso de prueba.

El segundo caso de prueba que se aplicará tendrá como objetivo vulnerar la base de datos que utiliza el SIGEB desde él mismo. Para lo anterior se introducirán sentencias SQL en los formularios buscando que el sistema responda trayendo datos confidenciales de la base de datos. Se llamaá a este caso de prueba SQLInjection haciendo alusión al nombre de la técnica que aplicaremos.

El siguiente caso de prueba tratará de vulnerar el sistema o de que éste falle aplicando la técnica de buffer overflow, la cual tendrá como objetivo revisar que el uso de cadenas de caracteres dentro del SIGEB se encuentre validado, de esta manera no se permitirá que el sistema falle al sobrecargar sus variables. El nombre de esté caso de prueba será el mismo que el de la técnica empleada.

Para el cuarto caso de prueba se introducirán datos que sean incompatibles con los tipos de datos que pida el formulario, es decir, se introducirán letras donde pida números y viceversa. Para este caso aplicaremos la técnica de formato de cadena, dejando con éste mismo nombre al caso de prueba.

Por último se aplicarán pruebas como desconectar el servidor donde se encuentra el SIGEB de la red para observar su comportamiento. De igual manera se intentará saltar el paso de autentificación como usuarios ingresando vía url. También se observara como responde el sistema si se deshabilita el uso de código javascript en el navegador.

5.3 Pruebas de caja negra.

Concepto de Caja Negra.

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.) Este comentario no obsta para que sean útiles en cualquier módulo del sistema.

Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del software son operativas
- La entrada se acepta de forma correcta
- Se produce una salida correcta
- La integridad de la información externa se mantiene.

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y determinación.

Resultados de las pruebas de Caja Negra.

Se aplicaran estas pruebas al formulario de acceso al sistema, que resulta el más importante en cuestiones de seguridad. Se aplicaran los casos más comunes para este ejercicio, como por ejemplo: dejar campos vacios y colocar información incorrecta o semicorrecta.

Figura 5.3.1. Formulario de acceso al SIGEB.

Como se puede observar, el formulario cuenta con dos campos para el acceso, en el primer caso de prueba se dejarán los dos campos vacíos obteniendo lo siguiente:

Figura 5.3.2. Formulario de acceso al SIGEB validado.

El resultado que arroja la prueba es que el SIGEB reacciona de la manera esperada al no permitir el acceso. Como un agregado nos informa que coloquemos la información.

Ahora se procederá a llenar un campo que será el de usuario y el de contraseña se quedará vacío.

Figura 5.3.3. Formulario de acceso al SIGEB validado.

El resultado es que de igual manera que en el caso anterior nos niega la entrada al sistema. Observamos que no nos menciona información extra en el campo de usuario.

Como siguiente prueba se introducirán datos en ambos campos, conociendo que los datos a introducir son incorrectos.

Figura 5.3.4. Formulario de acceso al SIGEB validado.

Se Observa que niega la entrada al sistema, haciendo del conocimiento del usuario que la información es incorrecta. Cabe destacar que no avisa cual de los datos es el incorrecto, previniendo que el usuario introduzca datos hasta saber que tiene alguno correcto. Mantiene el nombre del usuario solo por comodidad para que el usuario no lo vuelva a escribir, suponiendo que el dato incorrecto normalmente será la contraseña.

Ahora se colocará en el campo de contraseña un dato correcto con la finalidad de comprobar que el sistema niega el acceso si sólo se conoce alguno de los datos.

Figura 5.3.5. Formulario de acceso al SIGEB validado.

En la pantalla resultante se puede observar que el resultado es el mismo que si no se introdujera ningún dato correcto. Esto proporciona la confianza de que el sistema no ofrece ninguna información extra ante entradas de datos aleatorios.
Continuando con las pruebas de caja negra se tomará cualquier otro formulario del sistema para aplicarle algunas pruebas. El formulario al que se le aplicarán las pruebas es el siguiente:

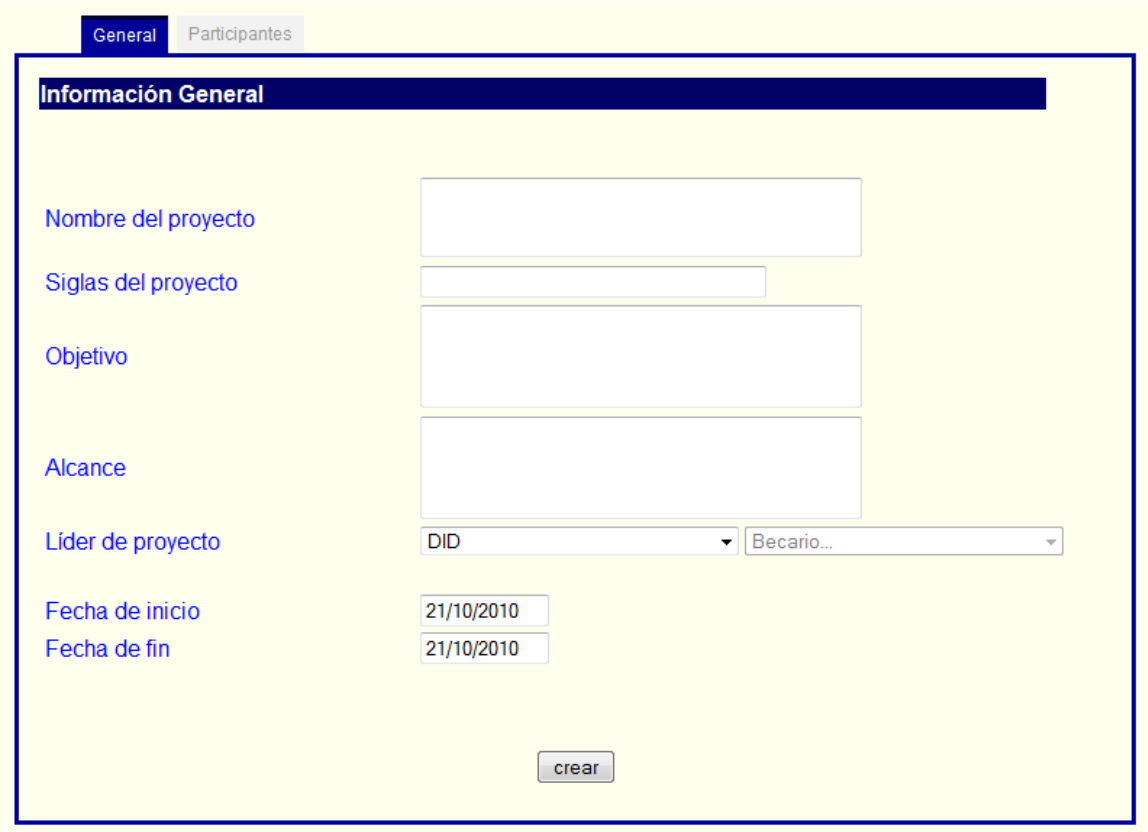


Figura 5.3.6. Formulario de creación de proyecto.

Podemos observar que este es un formulario con diferentes formas de ingresar datos. Tenemos cuatro campos de texto, dos campos desplegables dependientes y dos campos de fecha que automáticamente colocan la fecha del día en curso y no permite sean modificados directamente, para su modificación aparece un pequeño calendario donde se escoge una fecha válida.

Como primera prueba se tratará de crear un proyecto vacío, es decir, sin introducir ningún dato en el formulario.

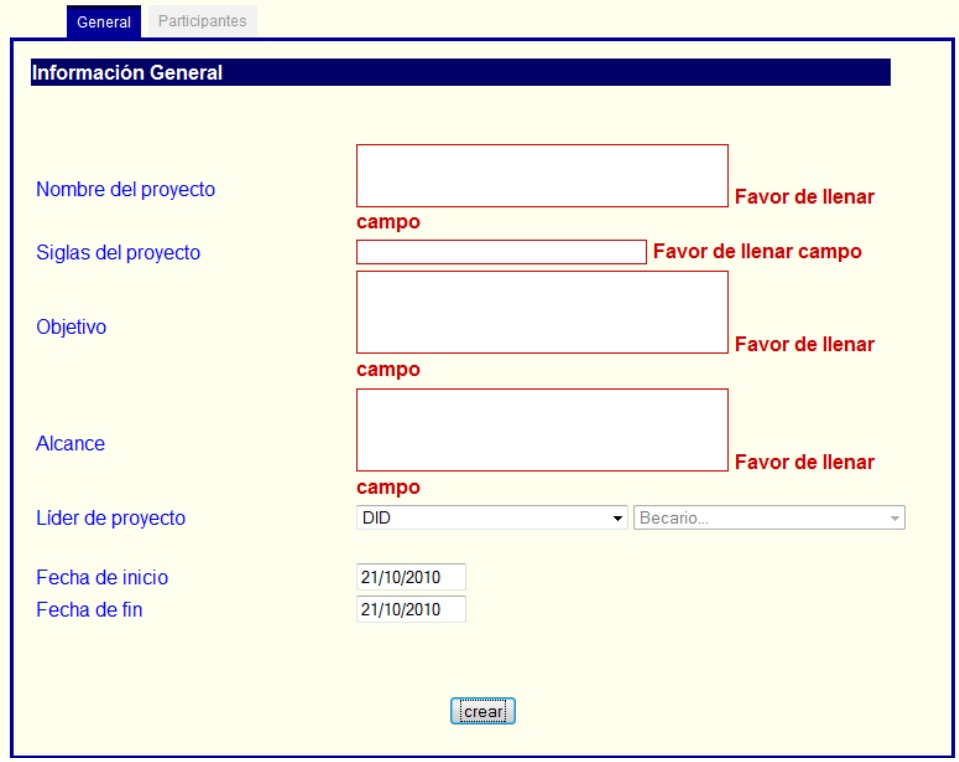


Figura 5.3.7. Formulario de creación de proyecto validado.

Se puede observar que al dejar todos los campos vacíos el SIGEB pide llenar los primeros cuatro campos, sin mencionar nada en los siguientes campos.

Se procede a llenar 3 de los campos que piden ser llenados para observar de qué manera reaccionará el SIGEB.

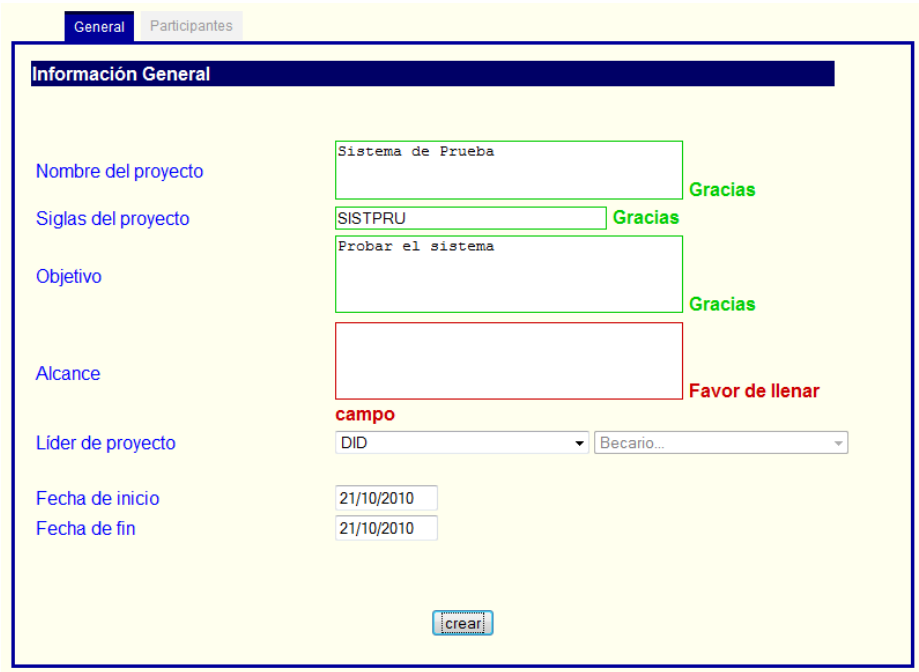


Figura 5.3.8. Formulario de creación de proyecto con validaciones.

Se nota que continúa pidiendo el ingreso de un dato. Para la siguiente prueba se llenarán los campos que nos marca como requeridos hasta el momento para observar si de esta manera crea el proyecto.

Figura 5.3.9. Validación del líder.

Se aprecia que en la parte superior de la pantalla se muestra una pequeña leyenda en rojo que nos comunica que es necesario escoger un líder en el proyecto.

Ahora se procederá a explicar los campos desplegables. El campo de la derecha se habilita al escoger un dato en el campo de la izquierda. Al escoger un dato en el campo de la izquierda el campo de la derecha adquiere los nombres en orden alfabético de los becarios del departamento escogido en el campo de la izquierda, dejando como valor por defecto al primer becario. En el caso

de los campos de fecha, pueden ser manipulados al gusto, es decir, puede crearse un proyecto anterior a la fecha actual, esto con la finalidad de poder documentar proyectos anteriores a la implementación del SIGEB en UNICA.

Por último se llenarán todos los campos del formulario esperando como respuesta la creación del proyecto.

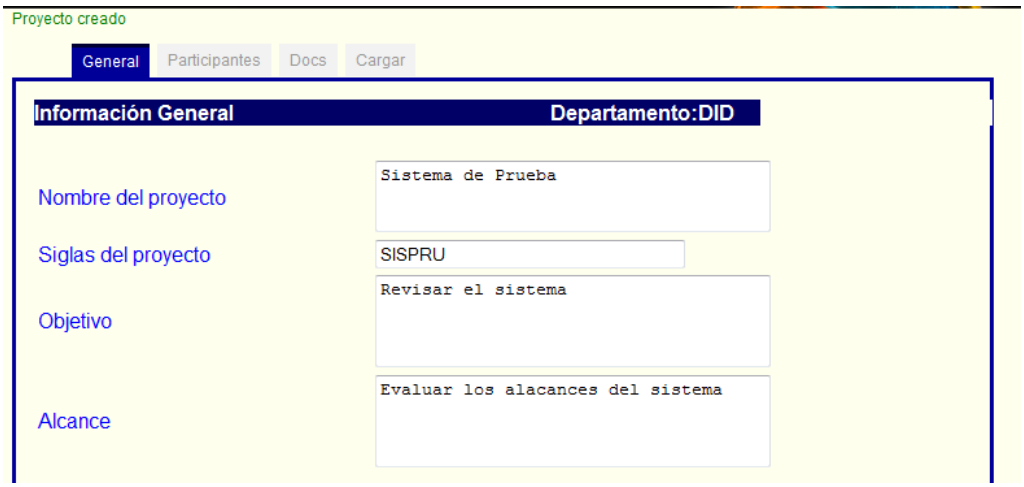


Figura 5.3.10. Creación del proyecto.

Aparece una leyenda en color verde en la parte superior de la pantalla que nos avisa que el proyecto ha sido creado. Con esta prueba se puede dar por concluido el proceso de pruebas de Caja Negra.

5.4 Pruebas de SQL Injection.

Concepto de SQL Injection.

Una inyección SQL o SQL Injection (en inglés) sucede cuando se inserta o "inyecta" un código SQL que actúa como código invasor dentro de otro código SQL, lo anterior con la finalidad de alterar su funcionamiento normal, y hacer que se ejecute maliciosamente el código invasor en la base de datos.

La inyección SQL es un problema de seguridad informática que debe ser tomado en cuenta por el programador para prevenirlo. Un programa hecho con descuido, displicencia, o con ignorancia sobre el problema, podrá ser vulnerable y la seguridad del sistema puede quedar ciertamente comprometida. Esto puede suceder tanto en programas ejecutándose en computadoras de escritorio, como en páginas Web, ya que éstas pueden funcionar mediante programas ejecutándose en el servidor que las aloja.

Al ejecutarse esa consulta por la base de datos, el código SQL inyectado también se ejecutará y podría hacer un sinnúmero de cosas, como insertar registros, modificar o eliminar datos, autorizar accesos e, incluso, ejecutar código malicioso en la computadora.

En el caso del Sistema de Gestión de Becarios (SIGEB) utilizamos una base de datos para almacenar la información que en el sistema de maneja. El SIGEB hace uso de una gran cantidad de sentencias SQL durante su funcionamiento normal, las cuales pueden ser cualquiera del Lenguaje de Manejo de Datos o DML por sus siglas en inglés.

La parte que se vuelve más vulnerable a ataques de este tipo es el formulario de ingreso al SIGEB, puesto que es el único formulario que puede ser visto por cualquier usuario con acceso a la red. Aplicaremos algunos ejemplos de SQL Injection con la finalidad de observar el comportamiento del SIGEB ante tal información.

En la figura 3.3.1 se muestra el formulario de ingreso que utiliza el SIGEB, se aplicará la prueba a este mismo formulario.

Resultados de las pruebas SQLInjection.

Normalmente una consulta SQL para validar un usuario en la base de datos se estructura de la siguiente manera: SELECT nombre, contraseña FROM tablaFulanita WHERE nombre='datoRecibido' and contraseña='datoRecibido2'; donde datoRecibido y datoRecibido2 son los datos que se ingresan en el formulario. La aplicación de SQLInjection en este caso funciona modificando los datos de entrada para completar alguna sentencia SQL válida. En el caso del formulario de ingreso al SIGEB se utiliza la siguiente forma de consulta: SELECT nombre, contraseña FROM tablaFulanita; después de esta consulta se valida el usuario en el resultado de la misma procurando no introducir datos de los campos directamente a la consulta. De esta manera se puede evitar una inyección de código SQL.

Para este caso se utilizará la cadena ' or '1'='1, esperando tener la siguiente cadena: SELECT nombre, contraseña FROM tablaFulanita WHERE nombre='datoRecibido' or '1'='1' and contraseña='datoRecibido2' or '1'='1'; observando que las condiciones serían automáticamente válidas, permitiendo el acceso.

Figura 5.4.1. Formulario de ingreso.

Se observa que el resultado de introducir la cadena es el esperado ya que no permite entrar al sistema. La razón es porque no se introdujo ningún dato directamente en la consulta, por lo que esta no puede ser modificada. Adicionalmente el sistema nos advierte que se introdujeron caracteres inválidos en la cadena, lo que quiere decir que el formulario solo acepta valores alfanuméricos, previniendo caracteres de escape u otros que ayuden a perforar la seguridad del sistema.

Ahora se colocará un usuario válido y en el campo de contraseña la cadena antes mencionada.

Figura 5.4.2. Formulario de ingreso validado.

El resultado es el esperado, no se permite el acceso. Aunque el SIGEB no responde con la leyenda de caracteres inválidos, también aplica las validaciones correspondientes al procesar la información.

Otro formulario que utiliza consultas de selección de datos utilizando información introducida por el usuario es el de buscar becario.

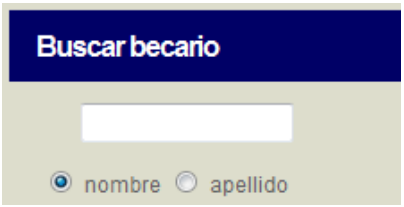


Figura 5.4.3. Formulario para buscar becario.

La query utilizada en este campo es: Select * from tabla where apellido like '%"parametro"%' and activo=1 and departamento="'depto'"; donde parámetro es el valor del campo.
Para tratar de vulnerar este formulario se introducirá la siguiente cadena: s"%' UNION SELECT becario FROM proyectobecario where rfc LIKE '%"P, tratando de hacer una unión entre un becario existente y un becario en algún proyecto.

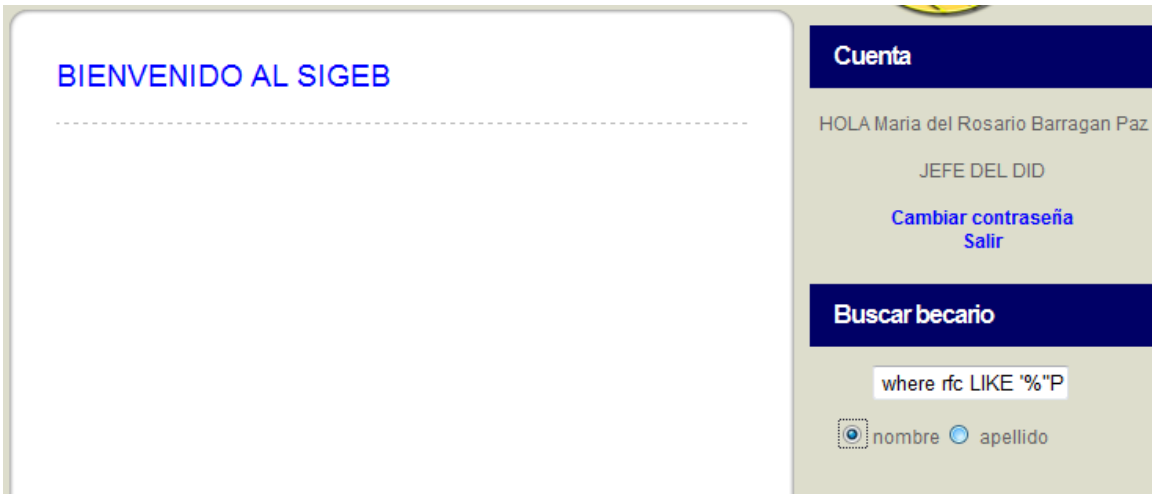


Figura 5.4.4. Formulario para buscar becario validado.

Como se puede observar en la figura el SIGEB no presenta ninguna información, por lo que se concluye que este formulario también se encuentra validado contra SQLInjection.

5.5 Pruebas de buffer-overflow.

Concepto de Buffer-Overflow.

Buffer-overflow o desbordamiento de buffer es un error de sistema causado por un defecto de programación, de tal forma que el programa que lo sufre pretende escribir más información en el buffer (unidad de memoria) de la que este puede alojar.

Este desbordamiento es posible porque el autor del programa no incluyó el código necesario para comprobar el tamaño y capacidad del buffer en relación con el volumen de datos que tiene que alojar. Los problemas comienzan cuando el exceso de datos se escribe en otras posiciones de memoria, con la perdida de los datos anteriores.

Si entre los datos perdidos por la sobreescritura se encuentran rutinas o procedimientos necesarios para el funcionamiento del programa que estamos ejecutando, el programa dará error. Cuando la memoria de un programa llega a sobreescribir en forma aleatoria, el programa generalmente fallará.

Resultados de las pruebas de Buffer-Overflow.

Esta prueba se aplicará en primer lugar al formulario de ingreso al SIGEB, se procederá a llenar ambos campos del formulario hasta su máxima capacidad y enviar la información para observar la respuesta. Se ingresan los datos como en la siguiente figura:

Figura 5.5.1. Formulario de ingreso.

Se puede ver que el formulario permite una capacidad de 15 caracteres para el usuario y de 20 caracteres para la contraseña. Enviamos los datos para observar el comportamiento del SIGEB.

Figura 5.5.2. Formulario de ingreso validado.

En el resultado se observa que nos niega el ingreso devolviendo una leyenda de información incorrecta. Con esta prueba se concluye que el formulario de ingreso al SIGEB tiene validadas las longitudes de cadena que utiliza.

El siguiente formulario a evaluar es el de búsqueda de becario, el cual se muestra en la figura 5.4.3. Se ingresará la cadena abcdefghijklmnopqrstuvwxyz para observar su comportamiento.

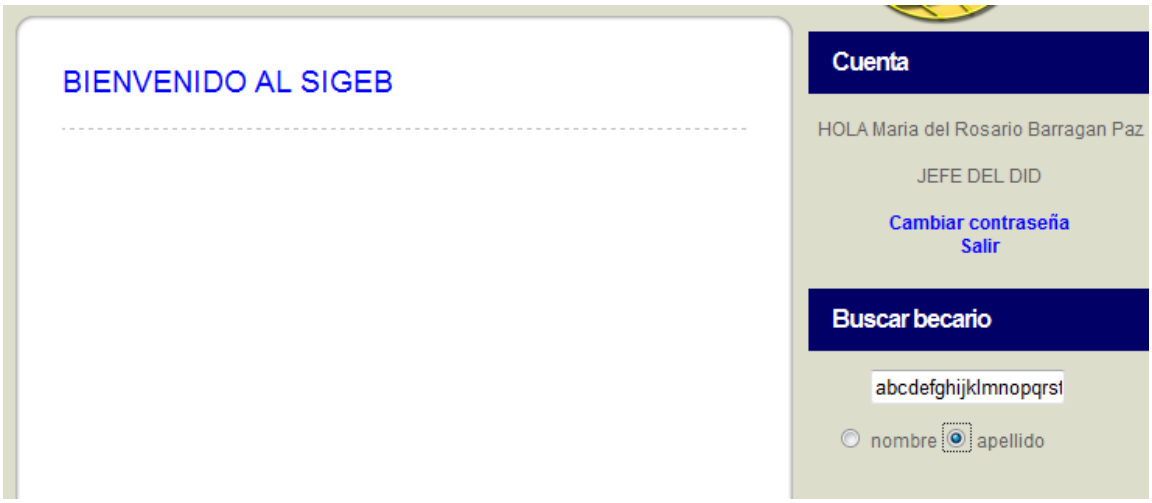


Figura 5.5.2. Formulario de búsqueda.

El resultado de buscar becarios se expresa en la parte inferior de la leyenda de bienvenido al SIGEB, como se puede observar no aparece ningún resultado ya que la cadena no es igual o similar a ningún apellido de algún becario, de igual manera no arroja ningún error al momento de hacer la consulta.

Por último se evaluará con esta prueba los formularios de creación de proyecto y edición de proyecto, el primero se muestra en la figura 5.3.2, el segundo se muestra en la siguiente figura.

Figura 5.5.3. Ventana de edición de proyecto.

Para esta prueba se colocarán cadenas de gran longitud en los campos que se requiere el ingreso de información. Lo anterior para observar cómo se comporta el SIGEB ante tal situación.

Para el caso del formulario de creación de proyecto:

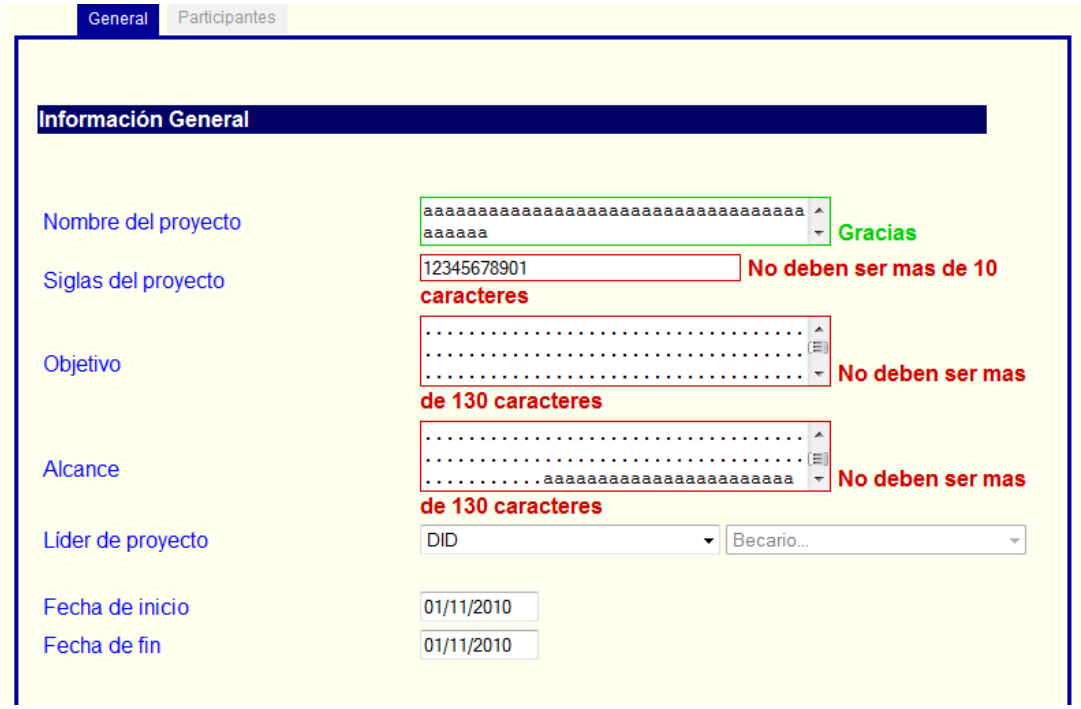


Figura 5.5.4. Ventana de creación de proyecto.

Se pueden observar todos los campos que reciben texto directamente del usuario se encuentran validados con una cierta longitud excepto el de nombre del proyecto, el cual puede recibir cualquier longitud de cadena ya que el espacio asignado en la base es ilimitado. Se dejó este campo sin validación para permitir nombres bastantes largos sin causar ningún conflicto.

Para el formulario de edición de proyecto:

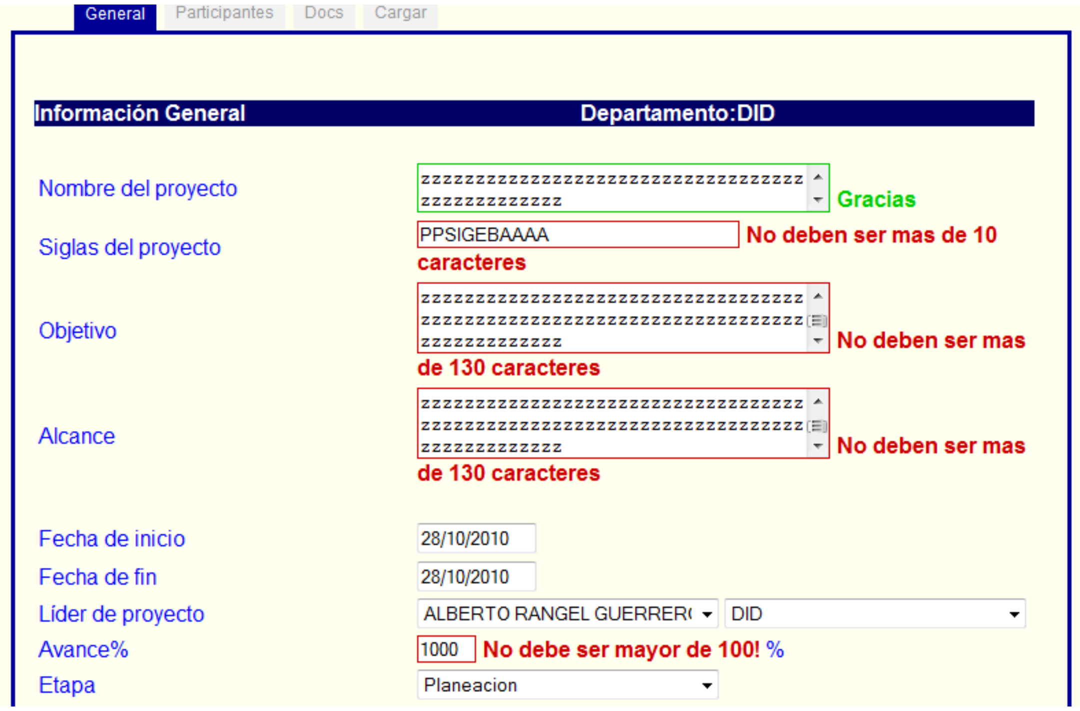


Figura 5.5.5. Ventana de edición de proyecto.

En este formulario se tiene un par de campos diferentes que en el de creación de proyecto, de los cuales solo uno recibe datos directamente del usuario y que como se puede observar se encuentra validado para datos no mayores de 100. Los demás campos tienen validaciones similares a las del formulario de creación de proyecto que no permiten más de cierta cantidad de caracteres.

5.6 Pruebas de formato de cadena.

Concepto de Formato de Cadena.

Los problemas de formato de cadena constituyen uno de los pocos ataques realmente nuevos que surgieron en años recientes.

Al igual que con muchos problemas de seguridad, la principal causa de los errores de formato de cadena es aceptar sin validar la entrada proporcionada por el usuario. Es posible utilizar errores de formato de cadena para escribir en ubicaciones de memoria arbitrarias, y el aspecto más peligroso es que esto llega a suceder sin manipular bloques de memoria adyacentes. Esta capacidad de diseminación permite a un atacante eludir protecciones de pila, e incluso modificar partes my pequeñas de memoria. El problema también llega a ocurrir cuando los formatos de cadena se leen a partir de una ubicación no confiable que controla el atacante. Este último aspecto del problema tiende a ser más frecuente en sistemas UNIX y Linux. En sistemas Windows las tablas de cadena de aplicación suelen mantenerse dentro del programa ejecutable o de las bibliotecas de vínculo dinámico (DLL, Dynamic Link Libraries) del recurso. Si un atacante reescribe el ejecutable principal o de las DLL, tendrá la posibilidad de realizar ataques mucho más directos que con errores de formato de cadena.

El formateo de datos para despliegue o almacenamiento tal vez represente una tarea un poco difícil; por tanto, en muchos lenguajes de computadora se incluyen rutinas para reformatear datos con facilidad. En casi todos los lenguajes la información de formato se describe a través de un tipo

de cadena, denominada formato de cadena. En realidad, el formato de cadena se define con el uso de lenguaje de procesamiento de datos limitado que está diseñado para facilitar la descripción de formatos de salida. Sin embargo, muchos desarrolladores cometen un sencillo error: utilizan datos de usuarios no confiables como formato de cadena; el resultado es que los atacantes pueden escribir cadenas en el lenguaje de procesamiento de datos para causar muchos problemas.

Resultados de las pruebas de Formato de Cadena.

Durante las pruebas anteriores se aplicaron implícitamente pruebas de formato de cadena al formulario de ingreso, por lo que ya no se documentará este formulario. Dado lo anterior comenzaremos aplicando esta prueba al formulario de búsqueda de becarios mostrado en la figura 5.4.1, donde la cadena esperada es del tipo alfabético, por lo que introduciremos números y signos de puntuación.

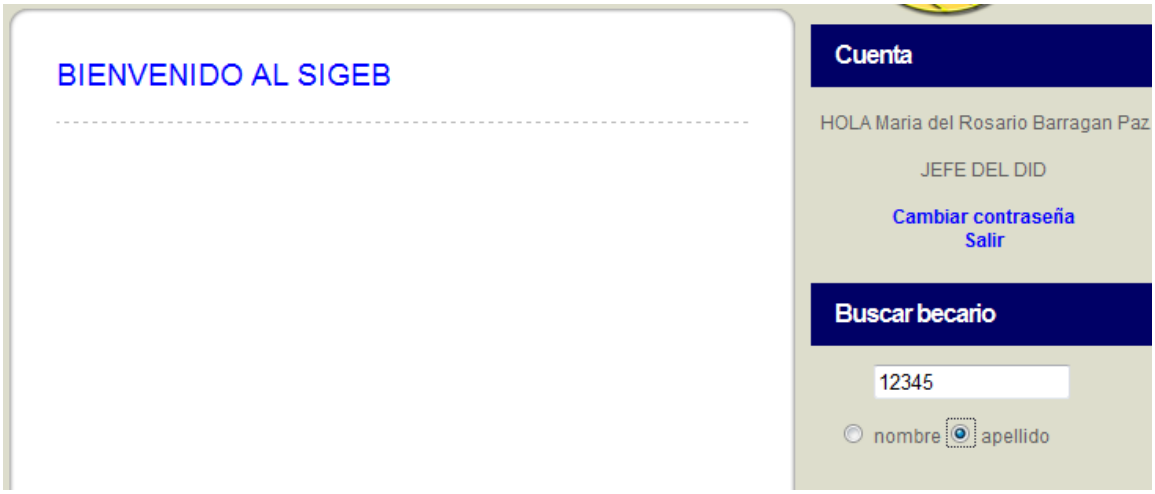


Figura 5.6.1. Formulario de búsqueda.

Para este primer caso se introdujerón números para la búsqueda observando que el resultado que arroja es nulo como se esperaba. Como ya se había mencionado, en la parte inferior de la leyenda de bienvenido al SIGEB es donde se arroja el resultado de la búsqueda, al encontrar el espacio vacío y no observar ninguna anomalía se llega a la conclusión de que el comportamiento fue el correcto.

Para el siguiente caso aplicaremos signos de puntuación sin caracteres alfanuméricos y combinando signos de puntuación con caracteres alfabéticos para observar el comportamiento.

Figura 5.6.2. Formulario de búsqueda validado.

En el resultado tampoco se observa anomalía alguna en ninguno de los casos, aunque no se muestran todos estos. El sistema tiene un comportamiento correcto ante las entradas no esperadas en el campo de búsqueda de becario.

El siguiente formulario a probar es el de revisión especializada de los departamentos de DSC y DROS, los cuales ocupan información numérica para su revisión. Los resultados son los siguientes:

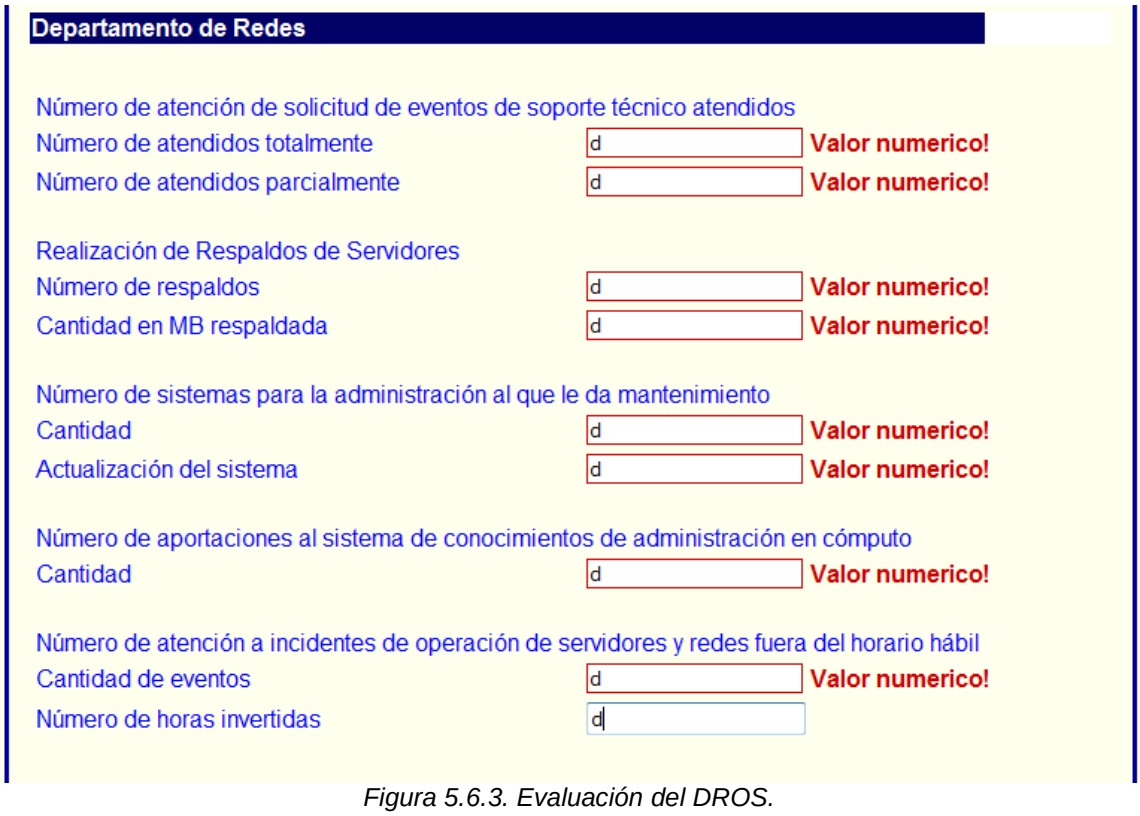


Figura 5.6.3. Evaluación del DROS.

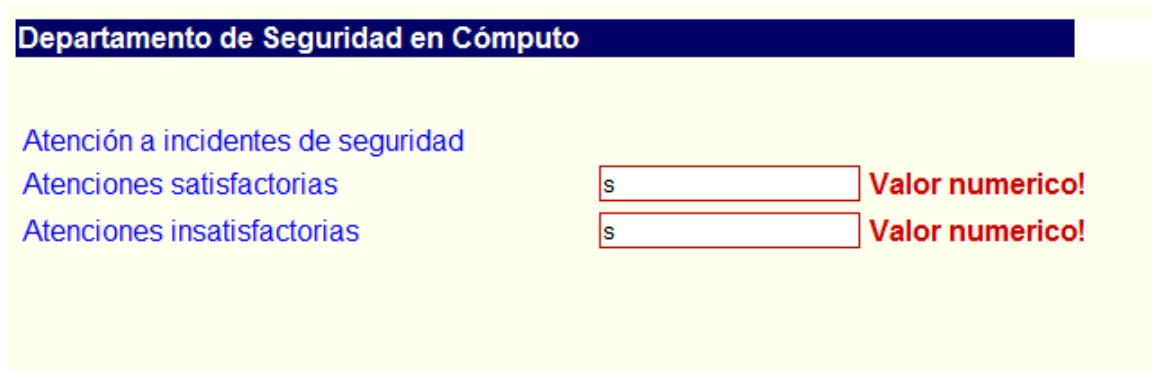


Figura 5.6.4. Evaluación del DSC.

Como se puede observar en ambas imágenes los campos avisan que requieren de valores numéricos en caso de que se coloquen otros tipos de datos no permitiendo su envío. Con esto se puede comprobar que las validaciones de los campos de estos formularios son las adecuadas, de esta manera el sistema no tiene que procesar información no adecuada.

En los formularios de creación y edición de proyecto los campos de texto permiten introducir cualquier tipo de información dado que las cadenas que se reciben pueden incluir signos de puntuación y caracteres alfanuméricos. El único campo que solo acepta números es el de porcentaje del proyecto, el cual se encuentra validado como se puede observar en la siguiente imagen:

Figura 5.6.5. Evaluación del DID.

Con la aplicación exitosa de la prueba a este último formulario se da por concluida la prueba de Formato de Cadena.

5.7 Otros casos de prueba.

Prueba de falla de conexión a la base de datos.

En esta prueba no se permitirá la comunicación entre el SIGEB y la base de datos que se encuentra en un servidor diferente, lo anterior con la finalidad de observar cómo responde el SIGEB ante la falta de la fuente de datos que alimenta al sistema.

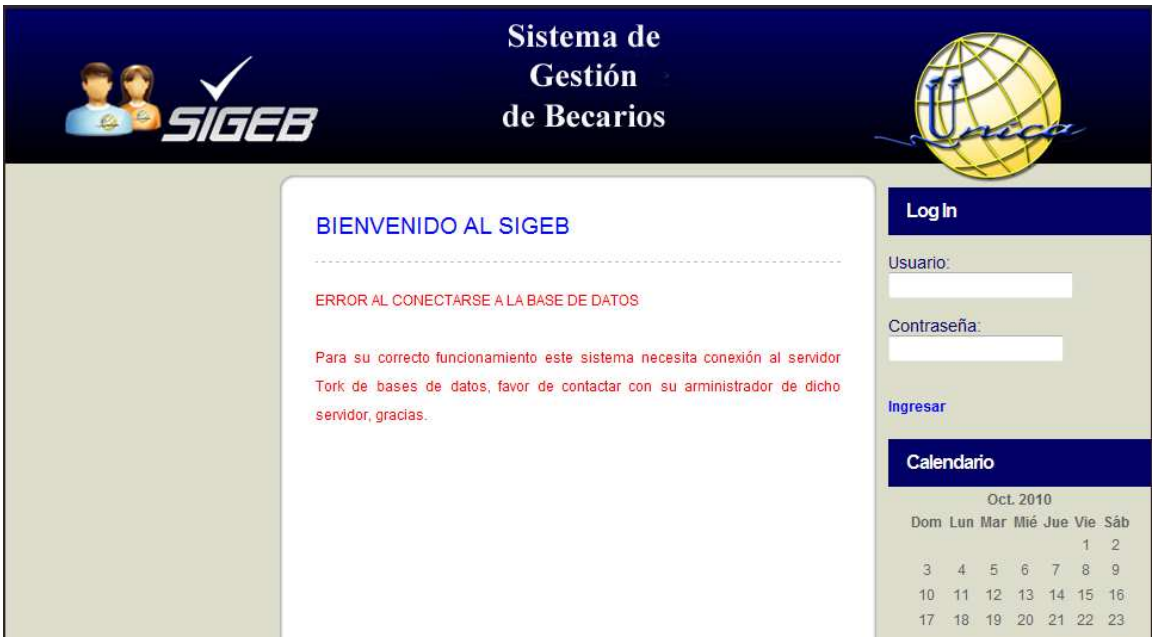


Figura 5.7.1. Falla al conectarse con la base de datos.

Como se puede ver en la figura anterior el SIGEB muestra un texto en color rojo avisando que la conexión entre éste mismo y la base de datos es incorrecta a la vez que pide contactar al administrador del servidor donde se encuentra. Con lo anterior podemos concluir que el sistema cuenta con validaciones de conexión a la base de datos.

Prueba de javascript deshabilitado en el navegador.

Deshabilitando Javascript en el navegador se pueden violar algunas reglas de seguridad escritas en este script de programación de algunos sistemas de información. Dado que el SIGEB hace las primeras validaciones con Java no tiene este problema, aún así se aplicaron algunas medidas de seguridad en el SIGEB para evitar que éste sea utilizado sin tener Javascript habilitado. A continuación se mostrará cómo responde el SIGEB ante tal situación.

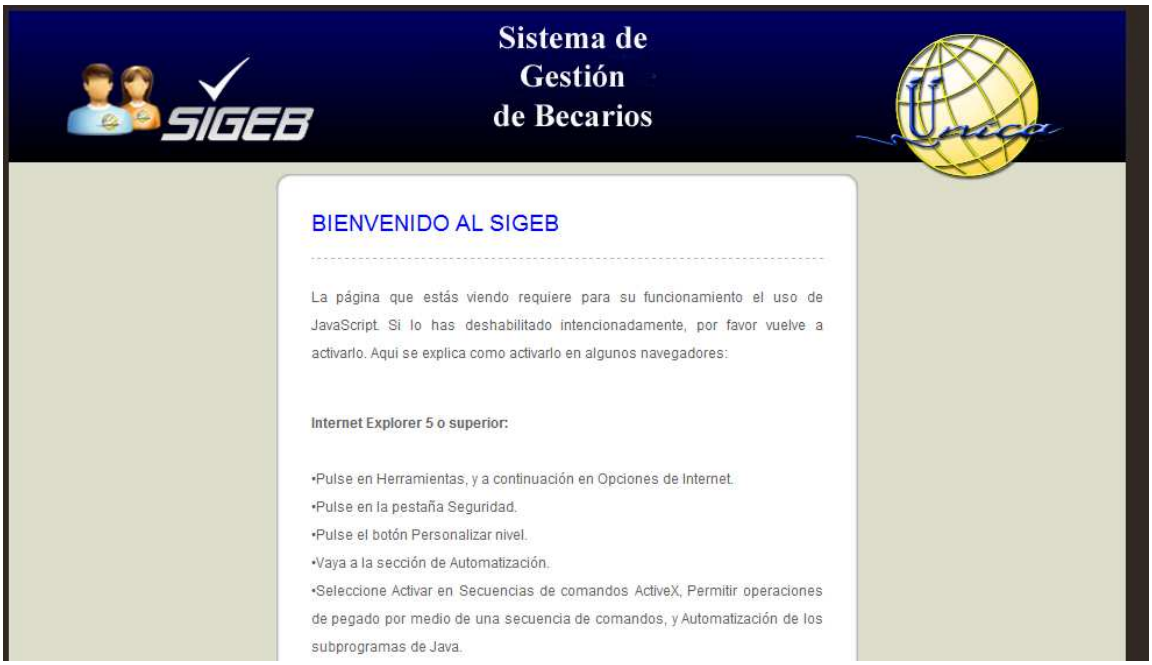


Figura 5.7.2. Error de del navegador.

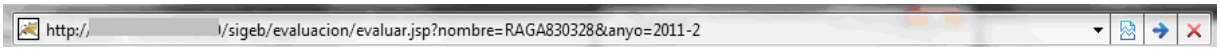
Al querer utilizar el SIGEB sin tener habilitado Javascript nos muestra una pantalla diferente advirtiéndonos que no tenemos habilitado javascript y que es necesario para su correcto funcionamiento. Adicionalmente nos da una breve explicación de cómo habilitarlo si no es del conocimiento del usuario. Con lo explicado anteriormente se puede concluir que el sistema está protegido contra ataques que pueden producirse con la deshabilitación de Javascript.

Prueba de ingreso al SIGEB por URL.

Muchos ataques comunes a sistemas que contienen sistema de validación al ingresar son violados vía URL, lo logran copiando o aplicando una URL válida que se use en el sistema una vez saltado el paso de la autenticación. Para este caso se usará la siguiente URL válida:

http://servidor/sigeb/evaluacion/evaluar.jsp?nombre=RAGA830328&anyo=2011-2

Para la aplicación se utilizará IExplorer 8. Se introduce la URL en la barra de navegación como sigue:



A continuación se presiona la tecla Intro o la flecha de Ir del navegador y se obtiene lo siguiente:

Figura 5.7.3. Formulario de ingreso.

Que es la pantalla de autenticación del sistema observando que en la barra de navegación se hace la modificación por la URL:

Que es la URL de inicio de sesión y es la que corresponde a la pantalla antes mostrada.

Se aplicó la misma prueba con otros navegadores comerciales obteniendo el mismo resultado en todos estos, por lo que se puede concluir que no es posible ingresar al SIGEB vía URL.