



**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INFORMÁTICA**

**“Mantenimiento – Práctica 2”**

*de Cózar Ruano, Fernando*

*Diezma Rodríguez, José María*

*Rivera Balseras, Christian*

*Gómez Herencia, Daniel*

*Piqueras López, Juan Ángel*

*Álvaro Díaz-Crespo, Miguel*

Grupo: 4  
Asignatura: Procesos de Ingeniería del Software  
Titulación: Grado en Ingeniería Informática  
Fecha: 12/12/2017

## Índice:

1-	Interfaz .....	2
a.	POSTMAN .....	2
b.	SELENIUM .....	3
c.	Soluciones y aportaciones .....	4
2-	VEGA .....	4
3-	OWASP ZAP .....	5
4-	KIUWAN .....	6
a.	Análisis Kiuwan antes de corregir ningún defecto .....	6
b.	Correcciones .....	8
i.	Confiabilidad .....	8
ii.	Mantenimiento .....	9
iii.	Seguridad .....	10
iv.	Eficiencia .....	11
c.	Segundo análisis con Kiuwan .....	11
d.	Tercer análisis con Kiuwan .....	11
5-	NUEVAS FUNCIONALIDADES .....	12

## 1- Interfaz

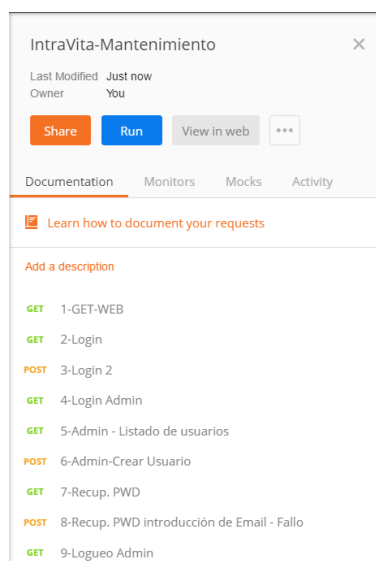
### a. POSTMAN

Como equipo de desarrollo, hemos utilizado la herramienta POSTMAN, además de Selenium, para realizar la fase de pruebas sobre la interfaz.

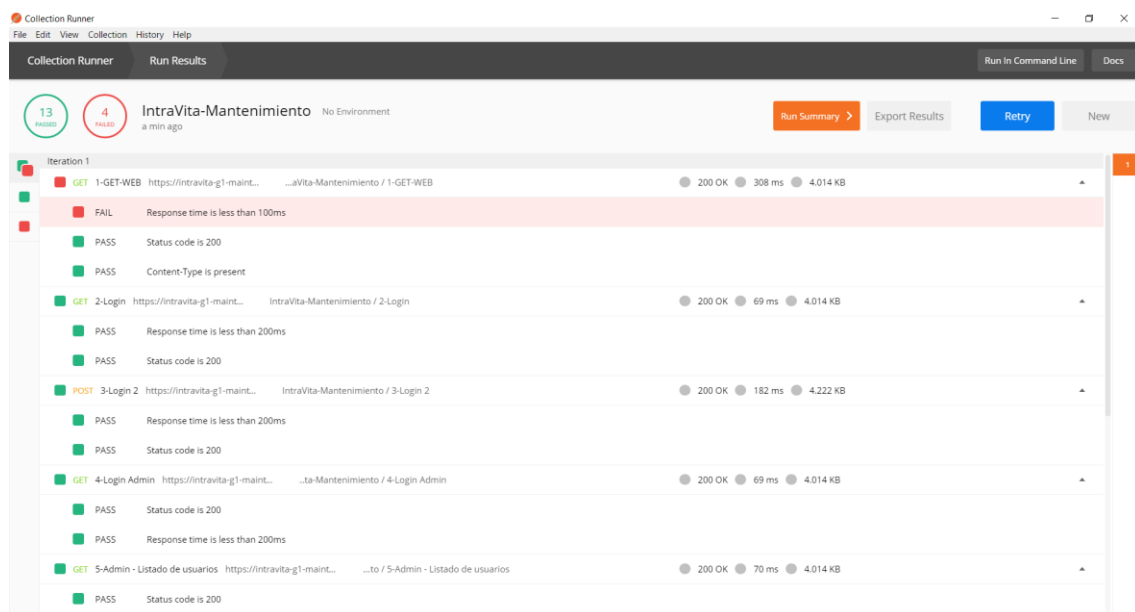
Las pruebas se han basado en realizar una comprobación de la correcta visualización de todas las ventanas de las que consta el proyecto, además de poner a prueba la mayoría de los formularios existentes.

En cada uno de los test que se han hecho, se ha añadido una línea de código que comprobará que los tiempos de espera son adecuados, nada más ponerse el equipo a trabajar con el proyecto, detectaron que éstos eran demasiado elevados.

A continuación, vamos a ver la colección de test:



La ejecución de la colección de pruebas da como resultado:



Observando los resultados, los tiempos de respuesta son muy elevados, debido a la gran cantidad de datos existentes en la base de datos y un código poco eficiente (detectado al poder ver el código a través de esta misma herramienta, se observa la gran cantidad de código que se genera en la propia vista).

Finalmente, procedemos a realizar el listado de las pequeñas deficiencias detectadas a lo largo de estas pruebas:

- A simple vista, observamos que falta el FAVICON, icono de 16x16 que decora las pestañas del navegador.
- Los botones no cuentan con ningún tipo de etiqueta “title” que oriente a los usuarios sobre las funcionalidades de cada uno de ellos, problema que no tendría demasiada importancia si existiera un apartado de “Ayuda” o similar, que tampoco existe.
- En las peticiones GET sobre las páginas, observamos que ciertos estilos CSS no son soportados, lo que podría derivar en incompatibilidades con otros navegadores (Sólo en lo referente a los estilos).
- Las modificaciones de las credenciales no reutilizan las exigencias de seguridad empleadas en el registro que pueden realizar los usuarios (desconocemos si era un requisito).

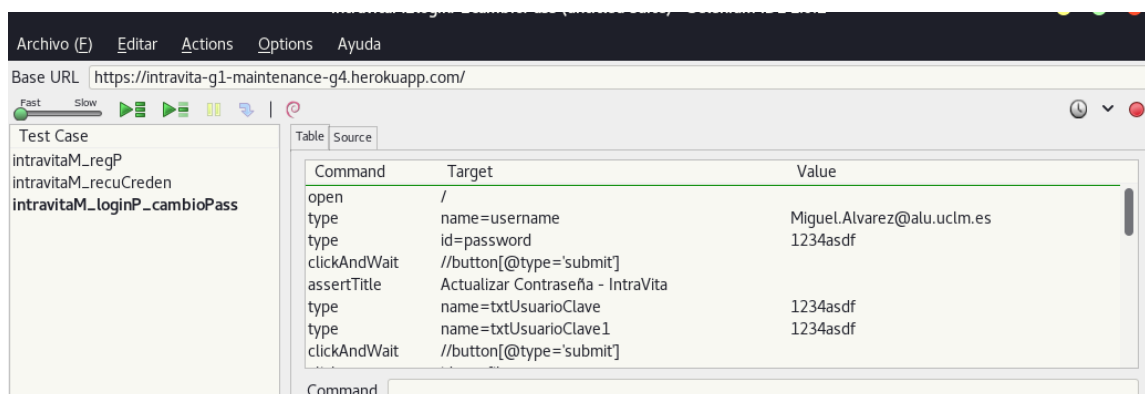
## b. SELENIUM

Desde el punto de vista de presentación, hemos realizado una serie de pruebas para encontrar posibles fallos entre ventanas o en elementos de la interfaz. Hemos utilizado Selenium para crear una serie de test para poner a prueba todos los elementos que componen la interfaz.

Las pruebas las hemos dividido en dos, distinguiendo entre el antes y el después de los cambios acordados en la reunión con el cliente (captcha).

En la primera parte se han realizado test donde se ha puesto a prueba las siguientes funcionalidades:

- Registro
- Login
- Normal
- Con actualización de contraseña
- Publicaciones: crear, editar y eliminar



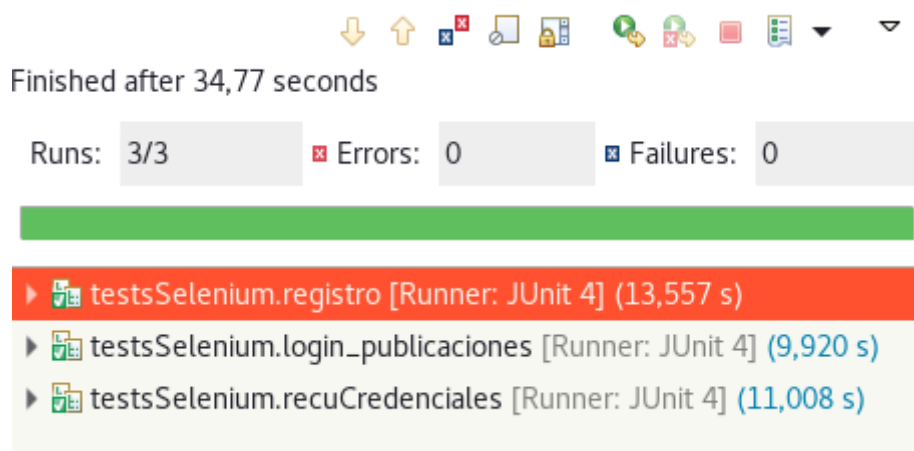
Todos los test han pasado correctamente. Sólo destacar que los tiempos de respuesta han sido muy elevados, defecto también detectado con Postman.

Dado que hemos tenido que desarrollar más funcionalidades, establecidas en la reunión de planificación, estos test han tenido que ser modificados.

Se han tenido en cuenta la utilización de **captchas** y **actualización de contraseñas**.

Con Selenium podemos automatizar los test, por lo también se puede comprobar si el sistema es seguro. Al añadir los captcha, Selenium no puede automatizarlos, por lo que automáticamente hacemos la aplicación más segura.

Al llevar a cabo los test, los anteriores no pasan dado que es imposible automatizar los captcha. Por ello se han modificado para tener en cuenta que no van a pasar.



### c. Soluciones y aportaciones

- Imágenes en las publicaciones ajustadas a las dimensiones del “*timeline*” del muro.
- Inclusión de Favicón (reutilizado de un proyecto anterior)
- Botón de ayuda
- Descripción de los botones

## 2- VEGA

Una de las herramientas que hemos utilizado para evaluar la seguridad del sitio web ha sido Vega, además de Kiuwan, y en su primer análisis, hemos detectado los siguientes problemas:

### Scan Alert Summary

<b>High</b>	(1 found)
Session Cookie Without Secure Flag	1
<b>Medium</b>	(None found)
<b>Low</b>	(3 found)
Form Password Field with Autocomplete Enabled	3
<b>Info</b>	(None found)

Básicamente, los fallos “Low” nos dicen que los formularios tienen habilitada la capacidad de autocompletar información.

Estos problemas se han centrado en los formularios de entrada al sistema (*login*) y el de registro.

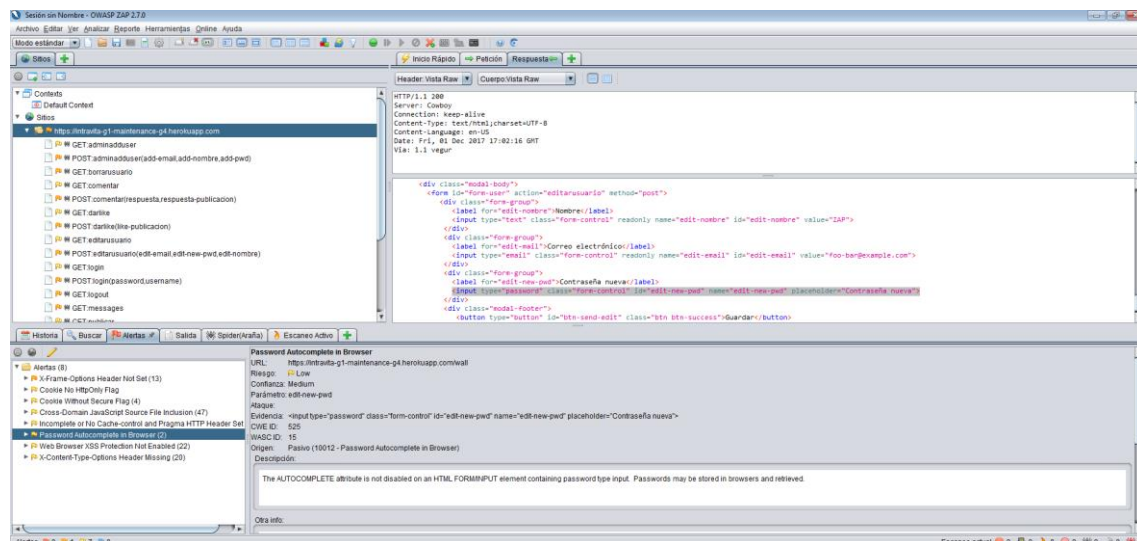
Una vez hemos detectado y corregido los fallos, el análisis final da el siguiente resultado:

#### Scan Alert Summary

<b>High</b>	(1 found)
Session Cookie Without Secure Flag	1
<b>Medium</b>	(None found)
<b>Low</b>	(None found)
<b>Info</b>	(None found)

### 3- OWASP ZAP

El análisis inicial (escaneo activo) es el siguiente:



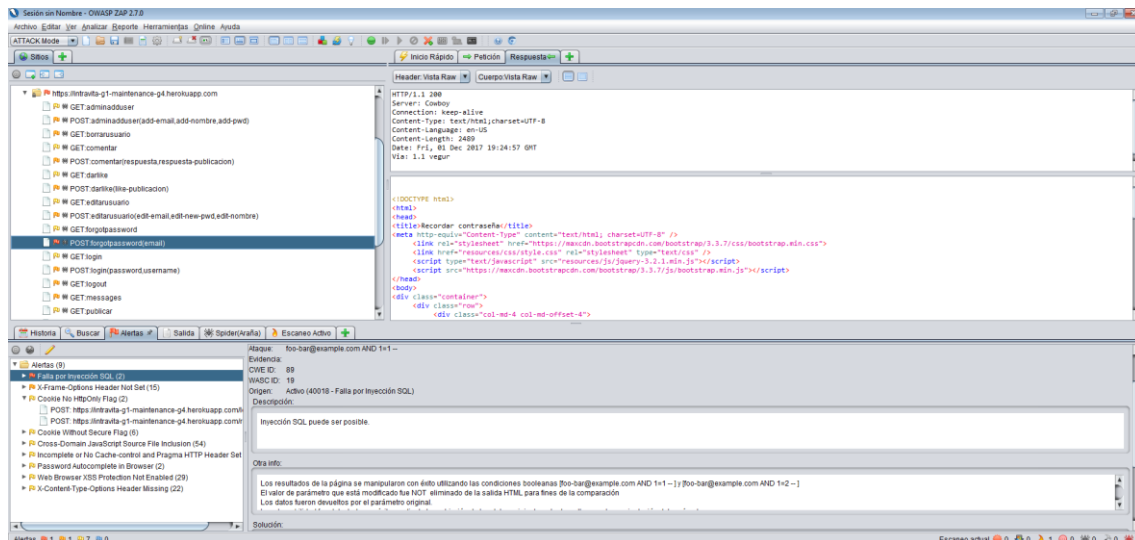
Observamos que sólo tenemos una alerta de consideración y 7 con menos importancia. Vamos a focalizar nuestra atención sobre la alerta más importante (señalada en naranja por la aplicación, riesgo medio).

La vulnerabilidad detectada proviene de la cabecera de las páginas, que no protege contra los ataques de “ClickJacking”. Tendremos que añadir la nueva cabecera a las páginas afectadas, también indicado por la aplicación.

Referencia para su solución: <https://developer.mozilla.org/es/docs/Web/HTTP/Headers/X-Frame-Options>

Las demás alertas, de una prioridad más baja, nos remarcen vulnerabilidades como el autocompletado de contraseñas en los formularios, como el indicado en la imagen anterior, otras relacionadas con el error en las cabeceras y finalmente, vulnerabilidades menores en las cookies.

Posteriormente, hicimos un análisis en modo “ataque” y a los resultados anteriores se ha añadido una nueva alerta, esta vez del nivel máximo de prioridad. Se trata de la posibilidad de que la aplicación sufra un ataque de *SQLinjection* como se muestra en la imagen:

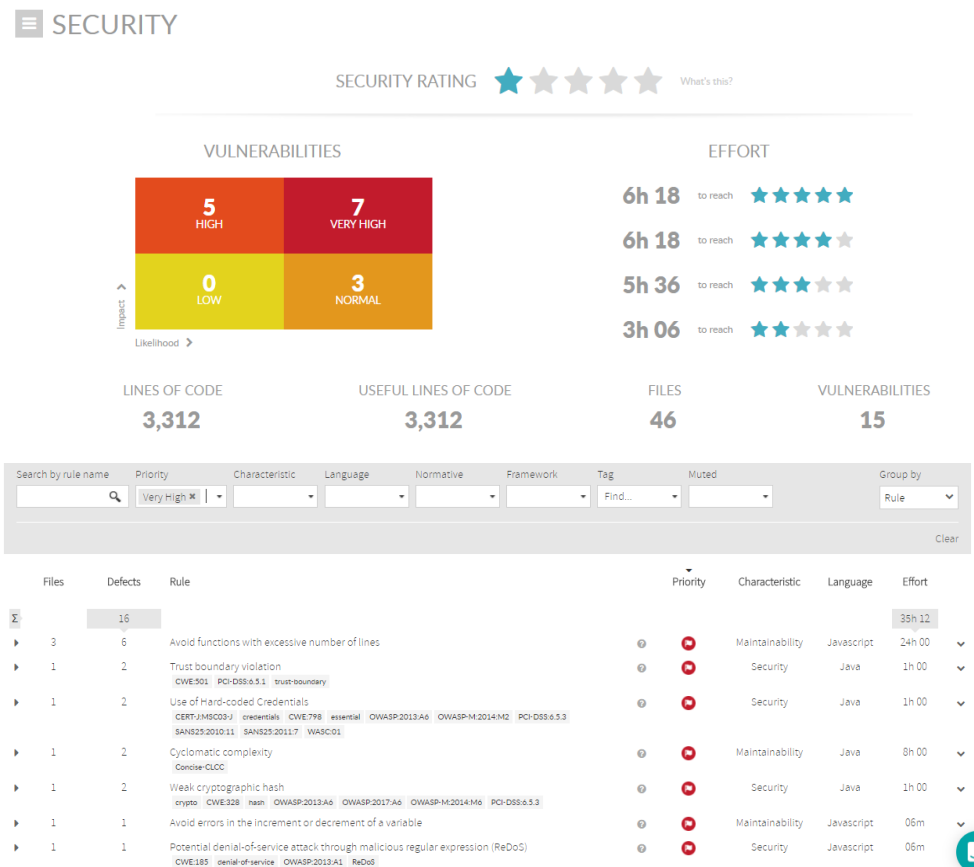


El problema por desgracias no tiene solución en el presente *Sprint* por falta de tiempo ya que requiere una serie de cambios importantes.

#### 4- KIUWAN

##### a. Análisis Kiuwan antes de corregir ningún defecto

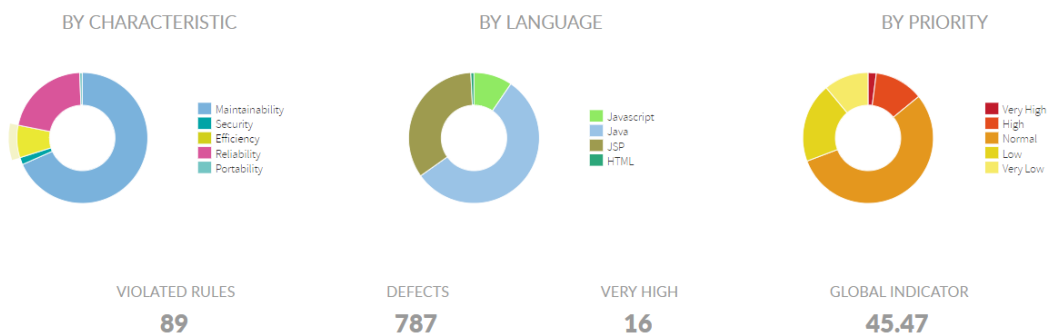
Como podemos ver en la siguiente imagen, tenemos 7 vulnerabilidades de nivel muy alto y 5 de nivel alto, que son las más importantes, con un total de 15 vulnerabilidades. Para alcanzar un nivel de seguridad de 5 estrellas nos indica que el esfuerzo es de 6 horas. Disponemos de 3.312 líneas de código



Se puede observar en los siguientes gráficos los defectos en base a los siguientes criterios:

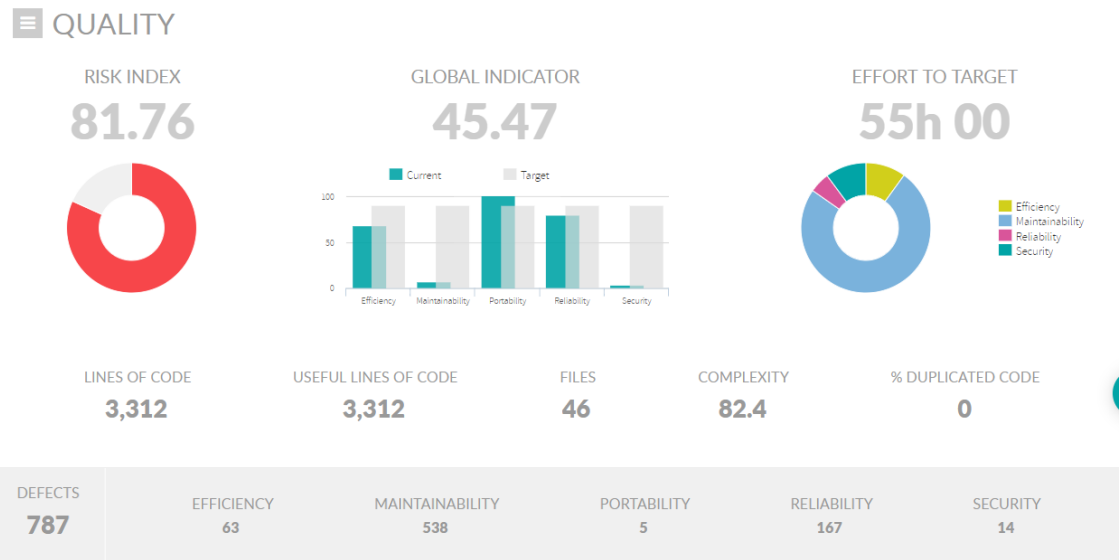
- Por características
- Por lenguaje
- Por prioridad

## DEFECTS



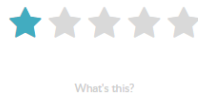
El índice de riesgo del proyecto es algo elevado, con un 81.76 %. El indicador global está por debajo del 50%, por lo que nos indica que se deben mejorar muchas cosas del proyecto, estimando el esfuerzo total para realizar las mejoras en 55 horas.





## FILES

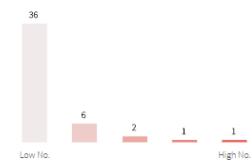
### SECURITY RATING



### BY SECURITY RATING



### BY NO. VULNERABILITIES



## b. Correcciones

### i. Confiabilidad

#### High

▼	1	1	Avoid calling non-final, non-static and non-private methods from constructors	●	🔴	Reliability	Java	30m	▼
▼		1	src/main/java/com/redsocial/modelo/Usuario.java						▼
			35   setNombre(nombre);						▼

Modificar el constructor de Usuario en el caso en el que se cree un Usuario solo con el campo del **Nombre**, ya que se puede hacer una simple asignación en vez de llamar a su propio método **set**.

▼	1	1	Avoid throwing RuntimeExceptions	●	🔴	Reliability	Java	4h 00	▼
▼		1	src/main/java/com/redsocial/auxiliares/SendMail.java						▼
			49   throw new RuntimeException(e);						▼

Eliminar la sentencia **"throw new RuntimeException"**.

## ii. Mantenimiento

### Very high

▼	1	2	Cyclomatic complexity Condse-CLCC	●	●	Maintainability	Java	8h 00	▼
▼		2	src/main/java/com/redsocal/controller/UsuarioController.java						▼
			68 public String borrarusuario(HttpServletRequest request, Model model) throws Exception {						▼
			165 public String admindeleteuser(@RequestParam String id, HttpServletRequest request, Model model) throws Exception {						▼

Resolver complejidad ciclomática: el problema surge a la hora de borrar usuarios (ambos métodos con el problema tenían el mismo fallo). Inicialmente lo que se hacía era recorrer TODOS los registros de usuarios, likes, mensajes... independientemente si son necesarios para borrar o no.

Esto se ha solventado modificando el método de borrado de usuarios, cuando borramos un usuario llamamos a otro método de borrar las publicaciones de ese usuario (ahora sólo recorreremos sus publicaciones y no todas), y para cada publicación borramos sus likes y respuestas (como lo anterior, ahora sólo recorreremos los registros necesarios y no todos, por lo que nos ahorramos la comparativa para saber si son útiles o no). Para finalizar, puesto que añadimos la funcionalidad de amistades, en el borrado de un usuario además borramos las relaciones de amistad que tiene y las solicitudes de amistad que ha enviado.

### High

▼	1	1	Provide Javadoc comments for public classes and interfaces Agile Alliance:Clear-CLDO Agile Alliance:Clear-CMTD Documentation	●	●	Maintainability	Java	30m	▼
▼		1	src/main/java/com/redsocal/auxiliares/DatosUsuario.java						▼
			3 public class DatosUsuario {						▼

Añadir comentario aclaratorio a la clase *DatosUsuario* para utilizarlo en el envío de email al recuperar la contraseña.

▼	3	3	Provide Javadoc comments for public classes and interfaces Agile Alliance:Clear-CLDO Agile Alliance:Clear-CMTD Documentation	●	●	Maintainability	Java	1h 30	▼
▼		1	src/main/java/com/redsocal/controller/PublicacionController.java						▼
			26 public class PublicacionController {						▼
▼		1	src/main/java/com/redsocal/controller/UsuarioController.java						▼
			28 public class UsuarioController {						▼
▼		1	src/main/java/com/redsocal/controller/WallController.java						▼
			34 public class WallController {						▼

Introducir comentarios en las clases indicadas que expliquen de forma clara que es lo que se pretende hacer.

▼	8	8	Avoid unused imports Agile Alliance:Condse-CDED CERT-JMSC56-J	●	●	Maintainability	Java	24m	▼
▶		1	src/main/java/com/redsocal/auxiliares/Utilidades.java						▼
▶		1	src/main/java/com/redsocal/controller/HomeController.java						▼
▶		1	src/main/java/com/redsocal/controller/PublicacionController.java						▼
▶		1	src/main/java/com/redsocal/controller/UsuarioController.java						▼
▶		1	src/main/java/com/redsocal/controller/WallController.java						▼
▶		1	src/main/java/com/redsocal/persistencia/DAOMensajesPrivados.java						▼
▶		1	src/main/java/com/redsocal/persistencia/DAOPublicacion.java						▼
▶		1	src/main/java/com/redsocal/persistencia/DAOUsuario.java						▼

Eliminar los import que no se usen de las clases correspondientes.

### iii. Seguridad

#### Very High

▼	1	2	Use of Hard-coded Credentials CERT-JMSC03-J credentials CWE-798 essential OWASP-2013:A6 OWASP-M-2014:M2 PCI-DSS-6.5.3 SANS25-2010:11 SANS25-2011:7 WASC01	🔍	🔴	Security	Java	1h 00	▼
▼		2	src/main/java/com/redsocal/auxiliares/SendMail.java						▼
		15	private static String userName = "atencion.cliente.dissw@gmail.com";						▼
		16	private static String password = "atencioncliente1234";						▼

Para la corrección de este defecto lo que hemos hecho ha sido crear una nueva clase *DatosUsuario* dentro del paquete *Auxiliares* en el que declaramos el nombre de usuario y la contraseña de la cuenta de la red social junto con los métodos get y set correspondientes. De esta forma quitamos esta parte de código de la clase de enviar email y obtenemos mediante los métodos get los datos necesarios en dicha clase.

▼	1	2	Weak cryptographic hash crypto CWE-328 hash OWASP-2013:A6 OWASP-2017:A6 OWASP-M-2014:M6 PCI-DSS-6.5.3	🔍	🔴	Security	Java	1h 00	▼
▼		2	src/main/java/com/redsocal/auxiliares/Utilidades.java						▼
		28	MessageDigest md = MessageDigest.getInstance("MD5");						▼
		53	MessageDigest md = MessageDigest.getInstance("MD5");						▼

Para la corrección de este defecto lo que hemos hecho ha sido eliminar las dos funciones que había para encriptar y desencriptar y hemos añadido la librería “*Apache Commons Codec*” en *Maven* de forma que hemos sustituido las llamadas a las funciones de encriptar y desencriptar anteriores por ***DigestUtils.md5Hex(contraseña)***. Este cambio lo hemos tenido que realizar en todas las clases en las que se hacía uso de la encriptación de claves.

▼	1	1	Potential denial-of-service attack through malicious regular expression (ReDoS) CWE-185 denial-of-service OWASP-2013:A1 ReDoS	🔍	🔴	Security	Javascript	06m	▼
▼		1	src/main/webapp/resources/js/password.js						▼
		54	/(^[\\w\\!\\#\\\$\\%\\&\\'\\*\\+\\-\\/\\=\\?\\^\\`\\{\\ \\}\\~\\.]*)?[\\w\\!\\#\\\$\\%\\&\\'\\*\\+\\-\\/\\=\\?\\^\\`\\{\\ \\}\\~\\.]*)?@((((([a-z0-9]{1}[a-z0-9\\-]{0,2})[a-z0-9]{1}) ([a-z]{1})\\.)+[a-z]{2,6}) (\\d{1,3})\\.){3}\\d{1,3})((...))						▼

Para solucionar el defecto de un posible ataque de denegación de servicio lo que hemos hecho ha sido eliminar la función que nos exponía, ya que en realidad no se estaba usando en ningún sitio.

#### High

▼	1	1	Standard pseudo-random number generators cannot withstand cryptographic attacks CERT-JMSC02-J CERT-JMSC63-J crypto CWE-330 CWE-338 OWASP-2013:A6 OWASP-M-2014:M6 OWASP-M-2014:M6 PCI-DSS-6.5.3 prng	🔍	🔴	Security	Java	30m	▼
▼		1	src/main/java/com/redsocal/controller/HomeController.java						▼
		91	int pin = (int) (Math.random() * (9999 - 1000 + 1) + 1000);						▼

Cambiar la forma en la que generamos el número aleatorio para recuperar las contraseñas. Utilizabamos un *Random* y ahora es *SecureRandom* por lo que genera un número más seguro que de la forma anterior.

#### iv. Eficiencia

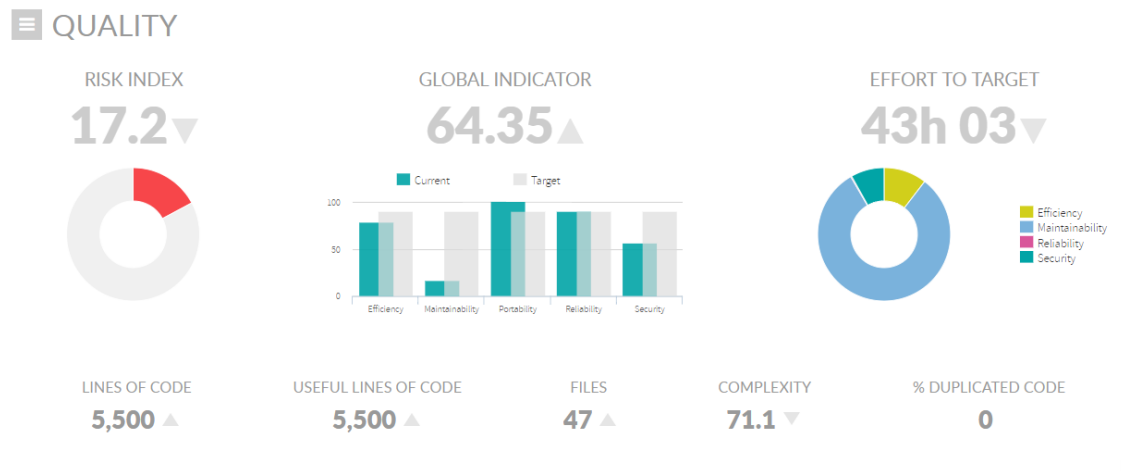
##### High

▼	3	8	Avoid using method calls in a loop	●	●	Efficiency	Java	4h 00	▼
▶		6	src/main/java/com/redsocial/controller/UsuarioController.java						▼
▶		1	src/main/java/com/redsocial/controller/PublicacionController.java						▼
▶		1	src/main/java/com/redsocial/controller/WallController.java						▼

Evitar las llamadas a métodos dentro de bucles. El problema detectado ha sido que dentro de bucles for se realizan llamadas al método `size()` para calcular el tamaño de un array, por lo que eso lo almacenamos en una variable fuera del bucle y en el propio bucle utilizamos dicha variable.

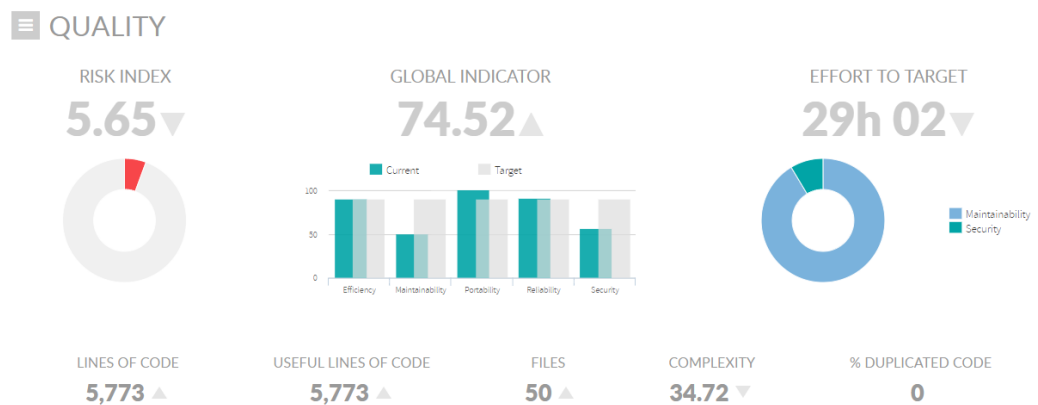
#### c. Segundo análisis con Kiuwan

En el segundo análisis, conseguimos una mejora considerable, bajando el índice de riesgo de un 81% a un 17% y como consecuencia, aumentando el indicador global de un 45% a un 64%, por lo que el número de horas necesarias para realizar las correcciones ha sido reducido.



#### d. Tercer análisis con Kiuwan

En el tercer análisis, corrigiendo más defectos, se ha vuelto a bajar el índice de riesgo de forma considerable hasta llegar a la cifra de un 5%, aumentando el indicador global en un 74%. Como mejora, tendríamos que centrarnos más en los defectos relacionados con la mantenibilidad y la seguridad.



## 5- NUEVAS FUNCIONALIDADES

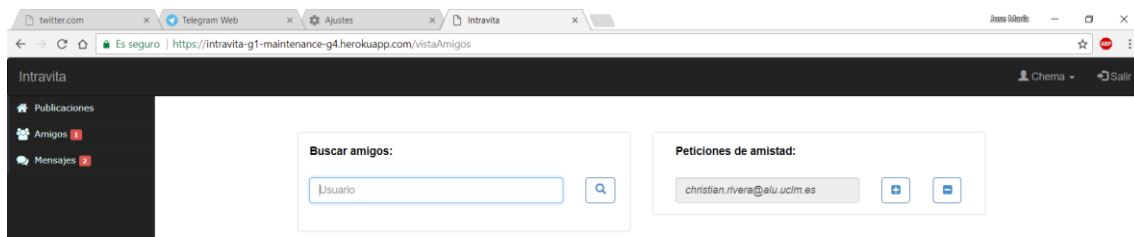
Entre las nuevas funcionalidades solicitadas por el cliente se encuentran la de las relaciones de amistad, implementación de una serie de captcha, y la obligación de cambiar la contraseña cada 4 horas (*posibilidad de ampliar este margen según las peticiones del cliente*).

### - Amistades:

Lo primero que hemos realizado, ha sido incluir el botón “Amigos” en el menú de la izquierda:

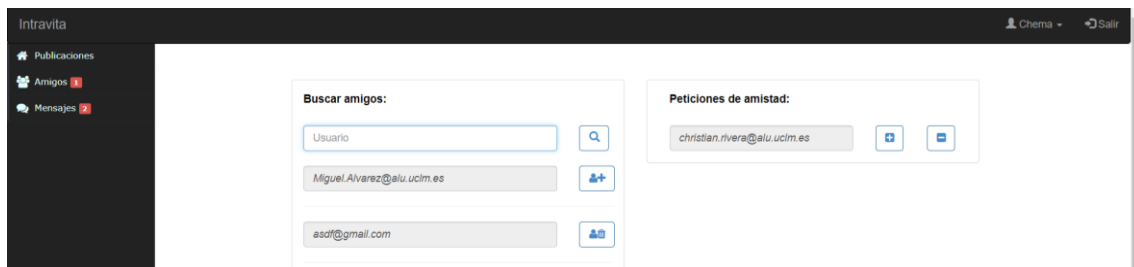


Una vez seleccionada la pestaña antes citada, veremos la siguiente pantalla:



Podemos observar dos apartados, el de buscar amigos, que tendrá un buscador para encontrar amigos (filtrando por email) y otro apartado con las solicitudes de amistad que tenemos pendientes, las cuales podrán aceptarse o no.

En caso de aceptarlas, al filtrar por amigos, podríamos encontrar a las personas que ya hemos aceptado, pero en lugar de tener un botón para agregarlo, al tenerlo ya, el botón será para eliminar a dicho usuario de nuestra lista de usuarios:



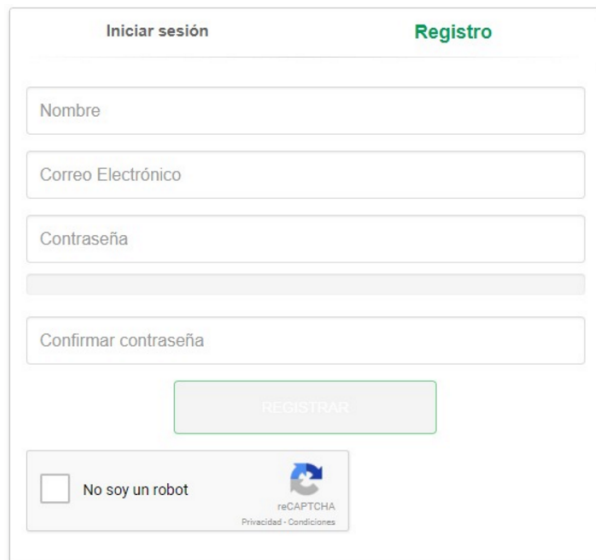
En la imagen podemos observar que el icono que aparece al lado de cada uno de los usuarios es diferente, debido a que uno de ellos ya forma parte de nuestra lista de amigos, por lo que sólo podremos eliminarlo de ella, y para quien todavía no es nuestro amigo, tendremos el botón de agregar (para aclarar la funcionalidad tenemos una descripción en cada uno de los botones).

### - Captcha:

Cada vez que un usuario tenga que iniciar sesión, cada vez que se registre, se use un pc nuevo o nos caduque la sesión tendremos la obligación de pasar por el captcha.



Mockup of the login form. It features two tabs: 'Iniciar sesión' (active) and 'Registro'. The 'Iniciar sesión' tab contains a 'Correo Electronico' input field, a 'Contraseña' input field, and a 'Entrar' button. Below the button is a 'Recordar Contraseña' link. At the bottom, there is a checkbox for 'No soy un robot' and a reCAPTCHA widget with links for 'Privacidad' and 'Condiciones'.



Mockup of the registration form. It features two tabs: 'Iniciar sesión' and 'Registro' (active). The 'Registro' tab contains a 'Nombre' input field, a 'Correo Electrónico' input field, a 'Contraseña' input field, a disabled 'Entrar' button, and a 'Confirmar contraseña' input field. Below these fields is a 'REGISTRAR' button. At the bottom, there is a checkbox for 'No soy un robot' and a reCAPTCHA widget with links for 'Privacidad' and 'Condiciones'.

#### - Actualización de contraseña:

Todos los usuarios que inicien sesión, pasadas cuatro horas desde el último inicio, tendrán la obligación de actualizar su contraseña. El cambio se producirá pasadas las citadas cuatro horas cuando vuelva a iniciar sesión:



Mockup of the password update form. It has a title 'Actualizar Contraseña'. Below the title are two input fields: 'Contraseña' and 'Repita la Contraseña'. At the bottom is a blue button with a refresh icon and the text 'Actualizar'.

#### - Publicaciones para amigos:

Se ha incluido la posibilidad mediante el botón de "Sólo Amigos" de realizar publicaciones que sólo serán visibles a los amigos del usuario en cuestión.



Mockup of the post creation form. It starts with the text '¿Qué estás pensando?'. Below this is a large text area for the post content. At the bottom left is a 'Seleccionar archivo' button, followed by the text 'Ningún archivo seleccionado'. Below these are two buttons: 'Solo amigos' and 'Publicar'.

- **Imágenes en las publicaciones:**

Las publicaciones cuentan con la posibilidad de incluir imágenes adjuntas:

