

# Validador de contraseñas

## Descripción

En una base de datos se tienen registrados los portales de acceso de y una serie de clientes junto con los datos de usuario y contraseña de cada cliente para los portales que posee. Estas credenciales caducan al cabo de un tiempo indeterminado, por lo que se debe realizar un proceso automático que diariamente revise que credenciales son válidas. En el caso que se encuentre una credencial invalida se debe enviar un correo al cliente solicitando las nuevas credenciales.

## Planteamiento del problema y tecnologías

Para dar solución a este problema lo más óptimo sería el uso de APIs públicas de sitios de internet para poder realizar la validación de usuario y contraseña. Sin embargo no todas las webs tiene APIs que podamos consumir. Siguiendo con el mismo espíritu de Web Scraping con el que dimos solución al problema de los televisores, directamente haremos la validación engañando a la web haciendo creer que estamos haciendo login desde la web.

Por requerimiento este desarrollo se va a escribir en Java y se usara un motor de bases de datos MySQL. Java es multiplataforma por lo que podremos ejecutar la herramienta independientemente del sistema operativo. Además, gracias a la gran comunidad de desarrolladores podremos importar ciertas dependencias que harán nuestro trabajo mas sencillo.

### Dependencias

- **jsoup** Nos permite crear un parser del HTML recibido.
- **JavaMailAPI** Librería nativa de Java(EE) que permite mandar emails.
- **MySQL Driver** Driver necesario para la comunicación con nuestra base de datos

## Detallando el problema y dándole solución

### Extracción de datos

Los usuarios se sacan de la base de datos en una sola Query que cruza las 3 tablas mostradas en el enunciado mediante INNER JOINS. De esta manera el resultado de la consulta mostraría todos los usuarios y sus respectivos usuarios en los portales. Mediante el INNER JOIN se excluyen aquellos usuarios que no tienen ningún usuario en ningún portal.

Con la consulta podremos crear un ArrayList de usuarios que seguiría el siguiente modelo:

### Class Cliente

- **int** id
- **String** nombre
- **String** email
- **ArrayList<Portal>** listaPortales

### Class Portal

- **int** id
- **String** nombre
- **String** username
- **String** password
- **String** loginURL
- **String** postURL

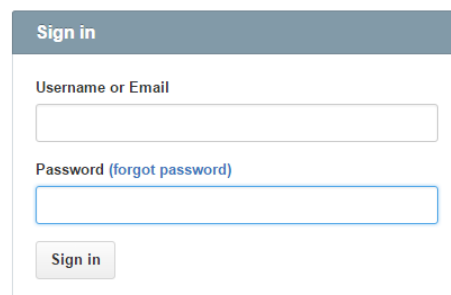
Una vez tenemos creada la estructura de datos sacada de la base de datos podremos iterar y realizar las operaciones que procedan por cada caso. Por motivos de tiempo solo se ha desarrollado el parser para Github.com.

### Parser de la web

El primer paso es descargar el HTML de la página de login del portal. Una vez descargado usaremos *jsoup* para poder encontrar rápidamente el HTML correspondiente al tag `<Form>`. Una vez lo hemos encontrado solo tendremos que leer todos los campos input y crear un string para realizar el POST.

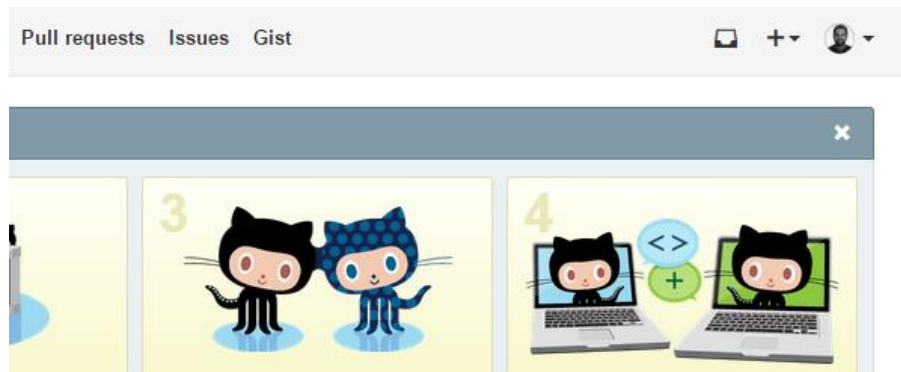
Dependiendo del resultado del POST nos mostrará HTMLs diferente. Las siguientes imágenes muestran un ejemplo de GitHub.com

Incorrect username or password.



The image shows a screenshot of the GitHub sign-in interface. At the top, there is a dark blue header with the text "Sign in" in white. Below the header, the form has two input fields: "Username or Email" and "Password (forgot password)". The "Password" field is highlighted with a blue border. Below the input fields is a "Sign in" button. Above the form, there is a red error message box that says "Incorrect username or password."

Error de validación GitHub



*Validación correcta en GitHub*

Como se puede observar será suficiente con buscar el String *“Incorrect username or password”*. Si hay match entonces la validación ha sido incorrecta, de lo contrario será correcto. Es necesario hacerlo así ya que aunque la validación sea incorrecta la respuesta sigue siendo un 200 OK.

### **Notas del programador**

Para generar la base de datos existe un script en la carpeta *“DbScript”*. La clase MySQLdb está parametrizada para añadir diferentes host, puerto, usuario o contraseñas.