Arquitectura de la aplicación distribuida segura (octubre 2019)

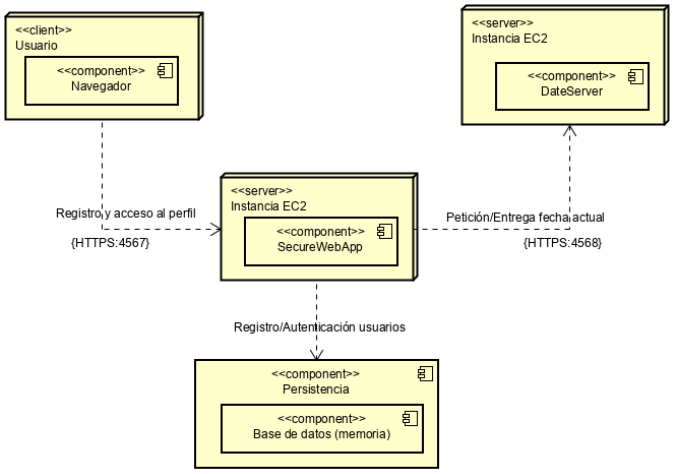
Carlos Andrés Medina Rivas – carlos.medina-ri@mail.escuelaing.edu.co
Escuela Colombiana de Ingeniería Julio Garavito

I. Introduction

Este documento tiene como objetivo mostrar la arquitectura, la implementación y las pruebas de una aplicación segura que permite el registro de un usuario, el acceso a su perfil ingresando con las credenciales y conectarse a un servidor que le muestre la fecha actual dentro del perfil. Ambos servicios se encuentran desplegados gracias a los servicios de AWS, en donde se realizaron las pruebas para verificar que los servidores intercambiaran certificados con el fin de garantizar integridad, autenticación y autorización.

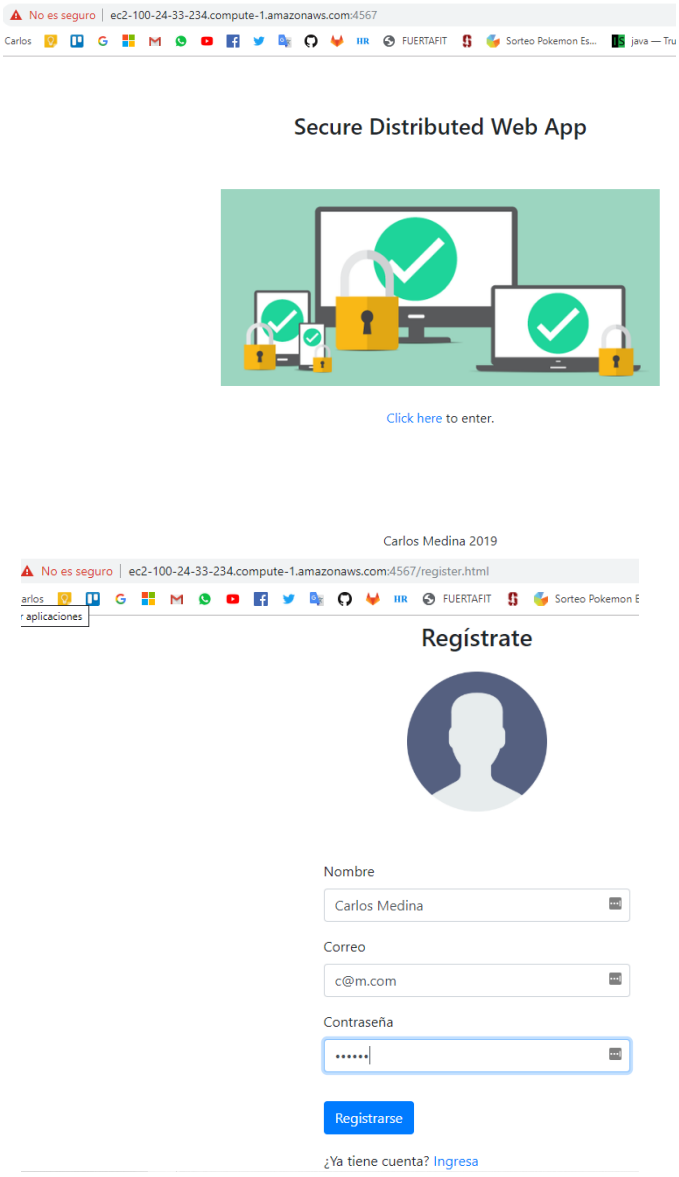
II. Arquitectura

El siguiente diagrama de despliegue que permite ver la arquitectura de seguridad usada para esta aplicación.

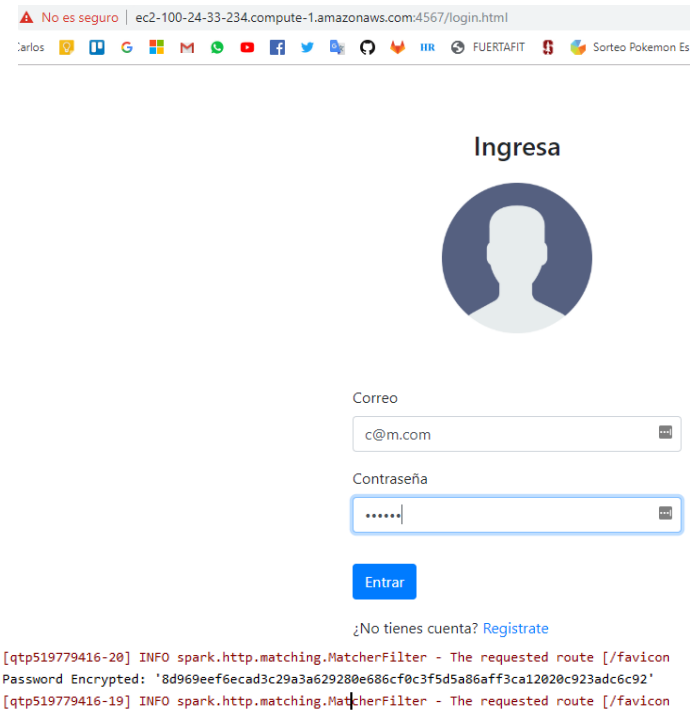


En el diagrama se pueden identificar los componentes Browser, los servicios del formulario y el servidor de la fecha ambos desplegados en EC. Junto al servicio del formulario está la persistencia que en este caso se realizó como almacenamiento en memoria usando una colección, haciendo las veces de base de datos para guardar a los usuarios que han sido registrados y así permitir el ingreso a la aplicación.

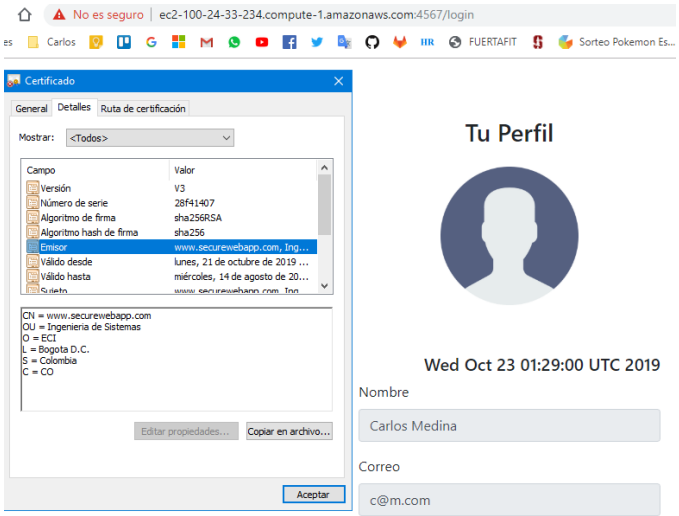
Cuando un usuario accede a la aplicación es necesario que acceda mediante el protocolo https para poder iniciarla. Inmediatamente aparecerá la página de inicio con la opción de ingresar (login) al formulario del perfil del usuario. Si este no tiene una "cuenta", desde el formulario del login es posible acceder al de registro. Aquí se le pedirán los datos como el nombre, correo y contraseña.



Cuando el usuario envía sus datos para el registro, la aplicación genera la cuenta con la contraseña ingresada cifrada y almacenándola de esa forma ofreciendo confidencialidad e integridad, ya que la contraseña original no puede ser visible al estar cifrada y cuando se va a ingresar, se comparan la contraseña acabada de escribir y la almacenada, que, al ser un hash, si un carácter no concuerda no permitirá dicho acceso.

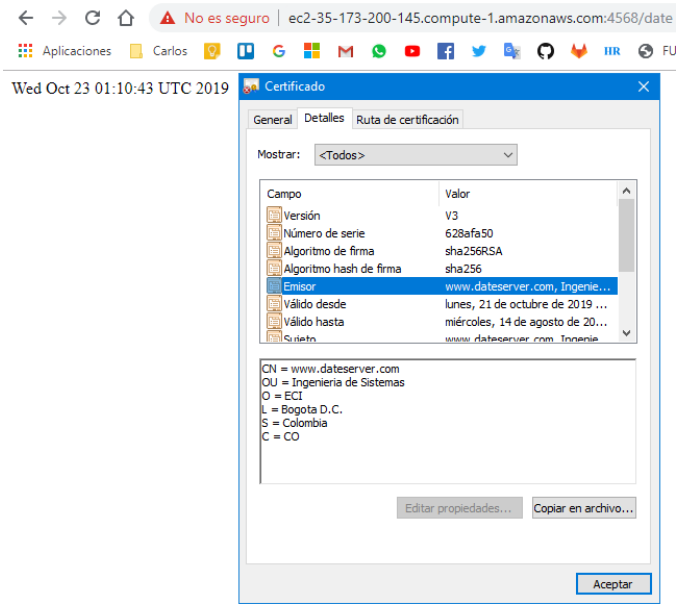


Aquí podemos ver otro criterio como el de autorización y autenticación, pues un usuario que no se haya registrado no se encuentra autorizado para acceder al perfil, y un usuario que ya tenga una cuenta puede autenticarse ante la aplicación para determinar que es quien dice ser.



Como se mencionó, las conexiones ente los servicios y el browser y el servicio son mediante https, de manera que se logre garantizar integridad, autenticación y autorización. Para esto, a cada aplicación se le generó un certificado mediante una keystore que almacena los certificados propios. A cada uno también se le generó un truststore que permite a cada servicio saber en quién confían. Esto permite el intercambio de certificados cuando se crea la conexión, logrando un canal cifrado entre dos partes.

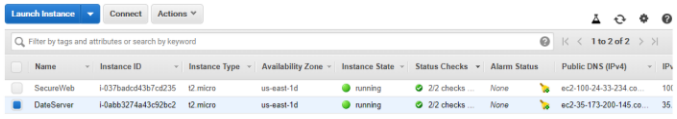
```
public class SecureWebApp {

    private static UserRepository userRepository = new UserRepository();

    public static void main(String[] args) {
        port(getPort());
        staticFiles.location( folder: "/public");
        secure( keystoreFile: "deploy/keyStoreSecureApp.jks",
                keystorePassword: "aACRSecure1App",
                truststoreFile: "deploy/truststoreDateServer.jks",
                truststorePassword: "jQYXDate5App");

static {
    //for localhost testing only
    javax.net.ssl.HttpsURLConnection.setDefaultHostnameVerifier(
            (hostname, sslSession) -> {
                if (hostname.equals("ec2-35-173-200-145.compute-1.amazonaws.com")) {
                    return true;
                }
                return false;
            });
}
```

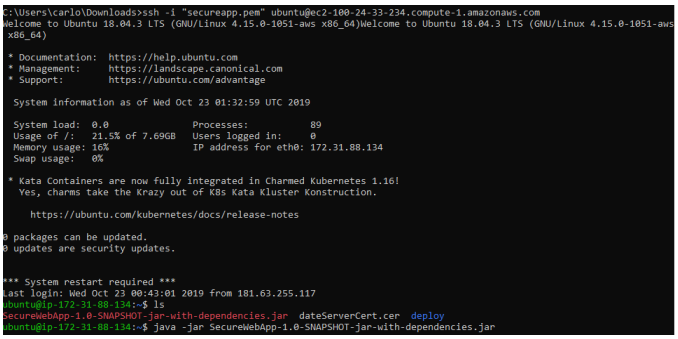


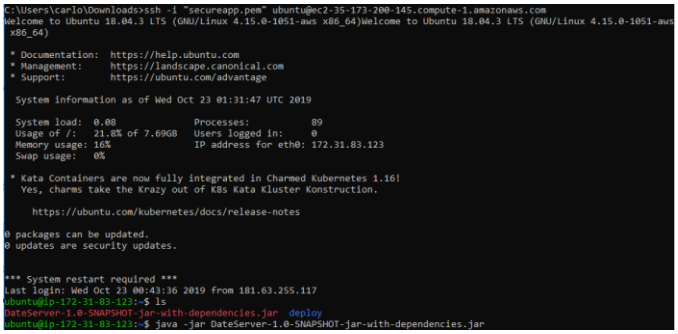
Cuando el usuario se autentica e ingresa a su perfil, la aplicación hace un llamado al servidor de la fecha para que este le devuelva dicho valor, el cual es mostrado en la misma vista que el perfil y cada vez que se recarga, se muestra la fecha y hora del momento.

```
private static String bringDate() {
    String date = "";
    URL url = null;
    try {
        url = new URL( spec: "https://ec2-35-173-200-145.compute-1.amazonaws.com:4568/date");
        BufferedReader br = new BufferedReader(new InputStreamReader(url.openStream()));
        String line = br.readLine();
        while (line != null) {
            date = line;
            line = br.readLine();
        }
    } catch (MalformedURLException e) {
        System.out.println(e.getMessage());
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
    return date;
}
```

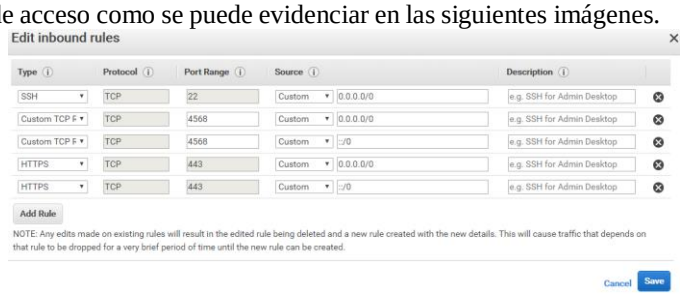
Ambos servicios se encuentran cada uno en una máquina virtual EC2. Para que las instancias ejecutaran las aplicaciones se realizó la conexión mediante ssh y sftp a cada instancia y se cargaron los archivos necesarios para la ejecución.

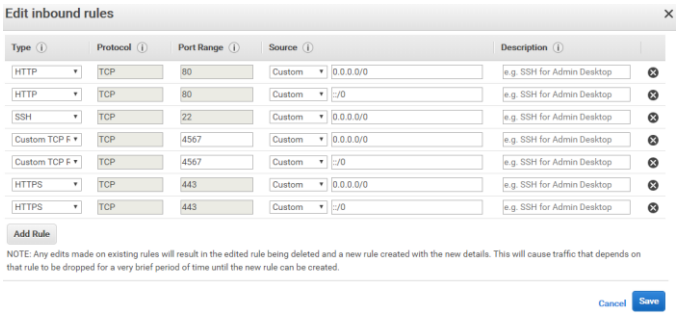




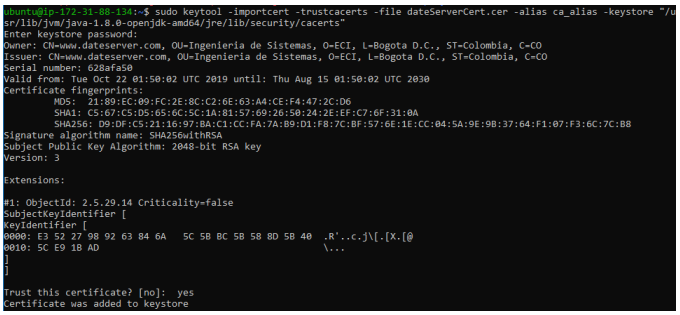


Para que se pudiera acceder vía https y el puerto establecido para spark desde las instancias, se tuvo que configurar las reglas de acceso como se puede evidenciar en las siguientes imágenes.





Finalmente, para permitir que la instancia donde se encuentra el formulario de registro y de inicio de sesión realizara la petición al servidor de fecha se debió agregar a los certificados de confianza de java, el certificado del servidor de fecha tal como se puede ver en la siguiente imagen..



III. Conclusiones

- Para lograr el intercambio de los certificados entre los servidores se deben tener en cuenta los archivos con los certificados de confianza y los propios, de manera que cuando se requieran sea posible autenticarse como servidor o como cliente, verificando que sea quien dice ser.
- Mediante el uso del protocolo https se activa el intercambio de certificados con las llaves públicas para así generar una conexión cifrada entre los servidores o el servidor y el browser para finalmente garantizar autenticación, autorización e integridad.
- Los métodos como el cifrado de las contraseñas hace que una aplicación mantenga la integridad y la confidencialidad sobre la información de los usuarios que, en este caso, son registrados en nuestra aplicación para garantizar autenticidad y autorización.

IV. Referencias

[1] Toby, «Trust Store vs Key Store - creando con keytool,» 14 Junio 2011. [En línea]. Available: https://www.it-swarm.net/es/java/trust-store-vs-key-store-creando-con-keytool/972567478/.

[2] tipsy, «spark-ssl,» 10 Mayo 2016. [En línea]. Available: https://github.com/tipsy/spark-ssl.

[3] «Documentation,» 27 Septiembre 2014. [En línea]. Available: http://sparkjava.com/documentation#getting-started.