Aplicación Distribuida Segura en Todos sus Frentes

Alejandro Toro Daza

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

Escuela Colombiana de Ingeniería Julio Garavito
14 de Marzo del 2021
Bogotá D.C.

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En la Aplicación Distribuida Segura en Todos sus Frentes se desarrolló una aplicación capaz de permitir un acceso seguro desde el browser a la aplicación. Es decir que este garantiza autenticación, autorización e integridad de usuarios, en los que se ha creado una interfaz de inicio de sesión que permite acceso seguro desde el navegador, y donde el usuario puede iniciar sesión de forma segura en el navegador. También se implementó la funcionalidad en la que al menos dos computadores se comunican entre ellos y el acceso de servicios remotos garantiza: autenticación, autorización e integridad entre los servicios, en los que nadie puede invocar los servicios si no está autorizado. La aplicación fue desarrollada con dos códigos fuente, el primero llamado LoginSeguro y el segundo llamado ServicioSeguro. En LoginSeguro el usuario puede iniciar sesión de forma segura, y este tiene su propio certificado HTTPS. ServicioSeguro se encarga de retornar una página también con su propio certificado HTTPS y su propia interfaz de usuario por aparte.

2. OBJETIVOS

- Implementar un programa capaz de manejar un inicio de sesión seguro utilizando el protocolo HTTPS.
- Elaborar los certificados para el inicio de sesión y el acceso autorizado seguro utilizando la herramienta **keytools** para la respectiva elaboración del certificado.
- Compilar y ejecutar el programa en Docker para su respectivo despliegue en la misma.
- Montar Docker en una máquina virtual utilizando AWS para así instalar Docker dentro de ella y poder desplegar el programa utilizando el servicio de la nube.

3. MARCO TEÓRICO

- **JavaScript:** Lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje.
- **CSS:** Lenguaje que define la apariencia de un documento escrito en un lenguaje de marcado (por ejemplo, HTML).
  Así, a los elementos de la página web creados con HTML se les dará la apariencia que se desee utilizando CSS: colores, espacios entre elementos, tipos de letra, ... separando de esta forma la estructura de la presentación.

Esta separación entre la estructura y la presentación es muy importante, ya que permite que sólo cambiando los CSS se modifique completamente el aspecto de una página web. Esto posibilita, entre otras cosas, que los usuarios puedan usar hojas de estilo personalizadas (como hojas de estilo de alto contraste o de accesibilidad).

- **HTML:** Lenguaje de marcado de hipertexto, y le permite al usuario crear y estructurar secciones, párrafos, encabezados, enlaces y elementos de cita en bloque (blockquotes) para páginas web y aplicaciones.
  HTML no es un lenguaje de programación, lo que significa que no tiene la capacidad de crear una funcionalidad dinámica. En cambio, hace posible organizar y formatear documentos, de manera similar a Microsoft Word.
- **Framework:** Conjuntos de componentes de software que se utilizan para crear la estructura de un software o una aplicación. Un framework puede verse como una caja de herramientas en la que el desarrollador busca los componentes necesarios. Este marco proporciona una estructura general para facilitar la labor de desarrollo.
  Los frameworks funcionan mediante el lenguaje de programación y desarrollan todo tipo de soporte: sitios web, juegos, aplicaciones móviles, etc. Sin embargo, también puede crear su propio marco de software.
- **HTTPS:** HTTPS (protocolo de Transferencia de Hiper-Texto) es un protocolo que permite establecer una conexión segura entre el servidor y el cliente, que no puede ser interceptada por personas no autorizadas. En resumidas cuentas, es la versión segura de el http (Hyper Text Transfer Protocol).

  Cómo funciona

  Una conexión HTTP estándar en Internet puede ser fácilmente secuestrada por partes no autorizadas. El propósito de una conexión HTTPS es evitar esto: encriptar los datos para asegurar una transmisión de datos segura. La transmisión está encriptada y el servidor autenticado.

  Cuando un usuario hace clic en un enlace o confirma una entrada de URL en la barra de direcciones con el botón Enter, el navegador establece una conexión. El servidor presenta un certificado que lo autentica como un proveedor genuino y confiable. Una vez que el cliente ha verificado la autenticidad, envía una clave de sesión que sólo puede leer el servidor. Sobre la base de estos datos clave, ahora se puede realizar el cifrado. Normalmente, se utiliza un certificado SSL.

  Ventajas

  Además de la ventaja obvia de la privacidad en línea, también hay otro pro. El uso de HTTPS no requiere ninguna instalación de software adicional. Esto significa que puede ser utilizado sin restricciones por cualquier persona. La autenticación con un certificado también inspira confianza en los clientes potenciales.

  Desventajas

  El HTTPS tiene algunas desventajas en comparación con las conexiones HTTP. Sin embargo, son muy pocas y deberían aceptarse como un compromiso por la seguridad que proporcionan.

- Hay cargos adicionales por certificados y costes crecientes con el aumento del tráfico. Estos pueden ser particularmente altos. Especialmente para sitios web nuevos y pequeños, estas tarifas pueden llegar a ser relativamente altas.
- Con conexiones HTTPS, el contenido no puede almacenarse en caché. Pero la tendencia hacia un mayor ancho de banda contrarresta esta desventaja.
- Un punto débil es también el menor rendimiento resultante del uso del cifrado SSL. El servidor debe realizar muchos más cálculos, aumentando así el tiempo de respuesta.
- Los hosts virtuales no funcionan con HTTPS.

Diferencia con HTTP

La principal diferencia es la seguridad. La tecnología es esencialmente la misma, pero HTTPS incluye encriptación SSL. Por lo tanto, en principio es posible establecer todo Internet con conexiones HTTPS. Sin embargo, debido a las desventajas antes mencionadas y por costumbre, casi nadie utiliza una conexión segura cuando no es absolutamente necesaria.

Seguridad

Dado que la diferencia con el HTTP es el uso de cifrado, la seguridad HTTPS depende únicamente de la técnica de cifrado utilizada. Actualmente se trata de SSL, que generalmente se considera segura. Sin embargo, debe tenerse en cuenta que una transmisión de datos segura por sí sola no es suficiente para protegerlos completamente, sino que también debe ser almacenada de forma segura por el destinatario.

- **SSL:** Acrónimo de Secure Sockets Layer (capa de sockets seguros), la tecnología estándar para mantener segura una conexión a Internet, así como para proteger cualquier información confidencial que se envía entre dos sistemas e impedir que los delincuentes lean y modifiquen cualquier dato que se transfiera, incluida información que pudiera considerarse personal. Los dos sistemas pueden ser un servidor y un cliente (por ejemplo, un sitio web de compras y un navegador) o de servidor a servidor (por ejemplo, una aplicación con información que puede identificarse como personal o con datos de nóminas). Esto lo lleva a cabo asegurándose de que todos los datos que se transfieren entre usuarios y sitios web o entre dos sistemas sean imposibles de leer. Utiliza algoritmos de cifrado para codificar los datos que se transmiten e impedir que los hackers los lean al enviarlos a través de la conexión. Esta información podría ser cualquier dato confidencial o personal, por ejemplo, números de tarjeta de crédito y otros datos bancarios, nombres y direcciones.
- **Docker:** Plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones

rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

La ejecución de Docker en AWS les ofrece a desarrolladores y administradores una manera muy confiable y económica de crear, enviar y ejecutar aplicaciones distribuidas en cualquier escala.

Docker le proporciona una manera estándar de ejecutar su código. Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

- **AWS:** Es un proveedor de servicios en la nube, nos permite disponer de almacenamiento, recursos de computación, aplicaciones móviles, bases de datos, entre otros.

  Amazon Web Services ofrece herramientas en las siguientes categorías:

  - **Cloud computing:** todo lo necesario para la creación de instancias y el mantenimiento o el escalado de las mismas. Amazon EC2 es el rey indiscutible dentro de los servicios de computación en la nube de Amazon.
  - **Bases de datos:** distintos tipos de bases de datos pueden permanecer en la nube mediante el servicio Amazon RDS, que incluye distintos tipos a elegir MySQL, PosgreSQL, Oracle, SQL Server y Amazon Aurora, o Amazon DynamoDB para NoSQL.
  - **Creación de redes virtuales:** permite la creación de redes privadas virtuales a través de la nube, gracias principalmente al servicio Amazon VPC.
  - **Aplicaciones empresariales:** Amazon WorkMail es el servicio de correo empresarial que ofrece Amazon, al que pueden unirse otros servicios como Amazon WorkDocs y Amazon WorkSpaces.
  - **Almacenamiento y gestores de contenido:** tipos de almacenamiento diferentes, tanto para archivos con acceso regular, poco frecuente o incluso como archivo. Amazon S3 es el servicio principal, aunque complementan la oferta otros como Amazon Glacier o Amazon EBS. En el siguiente vídeo (12:13 min.), puede verse un videotutorial con una definición general de AWS y una explicación del funcionamiento de Amazon S3:
  - **Inteligencia de negocios o Business Intelligence (BI):** sistemas para análisis de datos empresariales a gran escala y otros servicios para la gestión de flujos de datos.
  - **Gestión de aplicaciones móviles:** herramientas como Amazon Mobile Hub permiten la gestión, creación, testeo y mantenimiento de aplicaciones móviles a través de la nube.
  - **Internet de las cosas (Internet of Things, IoT):** para establecer conexiones y análisis de todos los dispositivos conectados a internet y los datos recogidos por los mismos.

- **Herramientas para desarrolladores:** para almacenar código, implementarlo automáticamente o incluso publicar software mediante un sistema de entrega continua.
- **Seguridad y control de acceso:** se pueden establecer autenticaciones en varios pasos para poder proteger el acceso a sus sistemas internos, ya estén en la nube o instalados de forma local en sus instalaciones.

4. DISEÑO Y CONSTRUCCIÓN

Para la creación del proyecto, se dispuso de dos programas, **LoginSeguro** y **ServicioSeguro**. **LoginSeguro** es programa que aparte de llevar a cabo la ejecución de todo el programa, contiene las llaves con sus respectivos certificados, las cuales las usa para usar el protocolo HTTPS y así poder mediante este protocolo redireccionar al HTML que contiene la interfaz de usuario final, donde el usuario puede iniciar sesión con sus respectivas credenciales, las cuales luego de iniciar sesión son redirigidas a otro HTML usando el protocolo HTTPS. Este programa contiene tres clases, que son **App**, **URLReader** y **User**. La clase **App** es la clase principal que se encarga de llevar a cabo la ejecución del programa, ya que tiene el **main**, donde se encarga del inicio de sesión haciendo uso de los certificados. La clase **URLReader** se encarga de realizar la respectiva lectura del URL de la llave ya creada con su respectivo certificado para así poder usar el protocolo HTTPS. Esta clase tiene todo lo relacionado específicamente con el certificado **myTrustStore**, generado a partir de la llave para poder almacenar los datos de forma mas segura. La clase **User** es una clase encargada de gestionar las credenciales del usuario, como lo son el correo electrónico y la contraseña. Esta clase se encarga única y exclusivamente de gestionar las credenciaes, utilizando métodos **get** y **set** para establecer tanto el correo electrónico como la contraseña del usuario en cuestión.
ServicioSeguro es un programa que se encarga de recibir peticiones del inicio de sesión del usuario, en la cual solo tiene una clase principal y encargada de la ejecución de todo el programa también llamada **App**, que se encarga de utilizar la llave con su respectivo certificado para darle el acceso al usuario luego de ingresar sus credenciales para así tener un acceso autorizado seguro, utilizando también el protocolo HTTPS.

Diagrama de Clases del programa LoginSeguro:

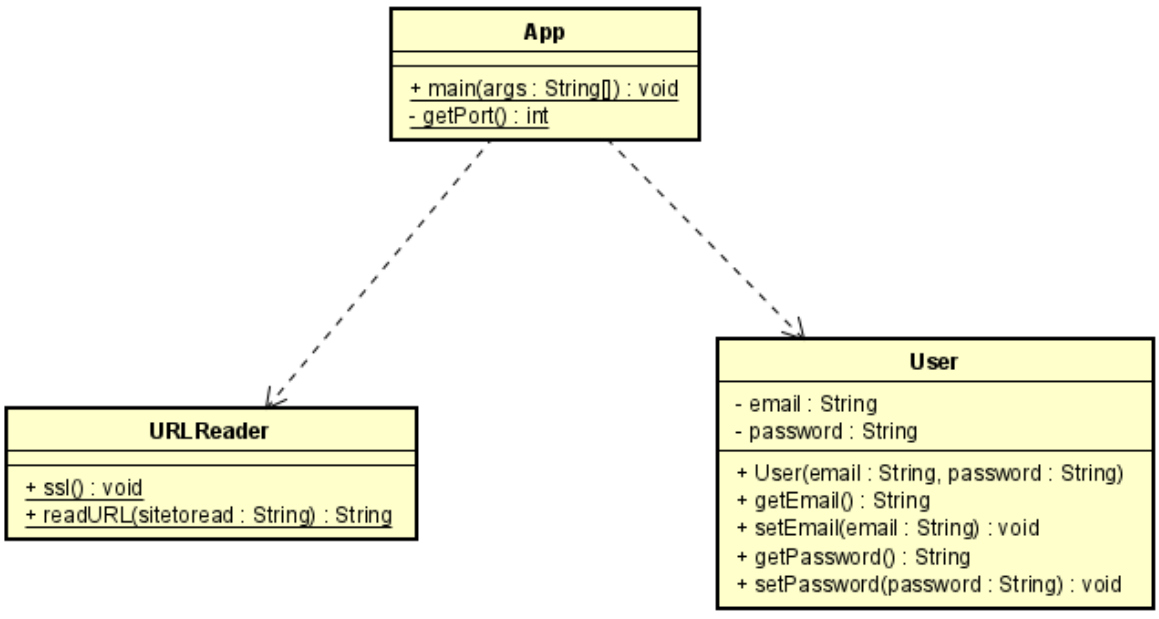
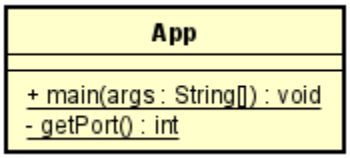


Diagrama de Clases del programa ServicioSeguro:



4.1. PRUEBAS

Pruebas en Maven

Para verificar que todo el código está funcionando con total normalidad, se implementaron las pruebas en **Junit**, las cuales fueron implementadas para cada uno de los programas **LoginSeguro** y **ServicioSeguro**. Para comprobar que primero las pruebas compilaron correctamente en **LoginSeguro**, se corrió el comando de Maven **mvn test**, el cual arrojó los siguientes resultados, demostrando que las pruebas fueron ejecutadas correctamente y que el código no contiene ningún tipo de error.

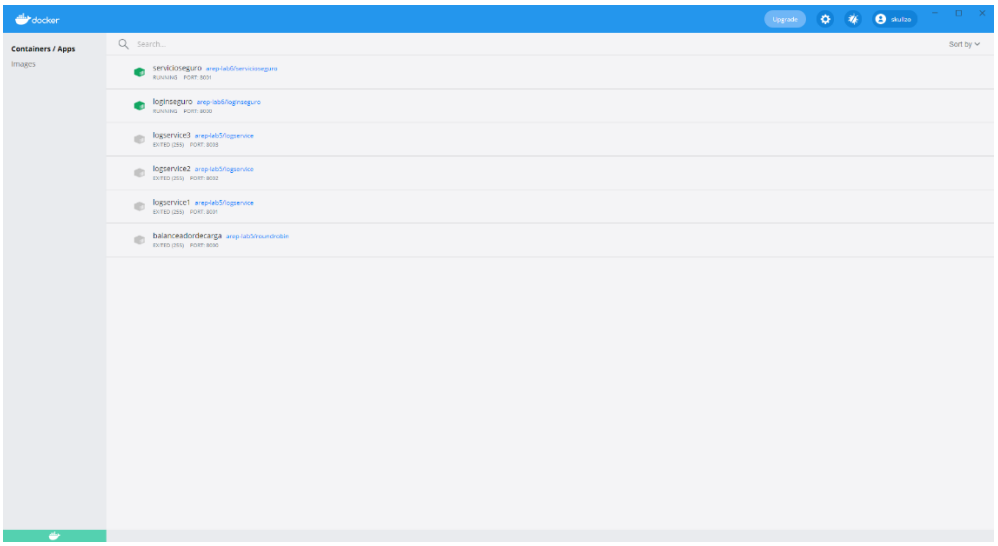
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.escuelaing.arep.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.031 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.287 s
[INFO] Finished at: 2021-03-10T14:12:19-05:00
[INFO] ------------------------------------------------------------------------
```

Luego, se realizó exactamente el mismo procedimiento, pero esta vez con el programa **ServicioSeguro**, en el que se ejecutó el mismo comando de Maven **mvn test** y arrojó también resultados exitosos, demostrando también que el código de **ServicioSeguro** y las pruebas fueron realizadas correctamente.

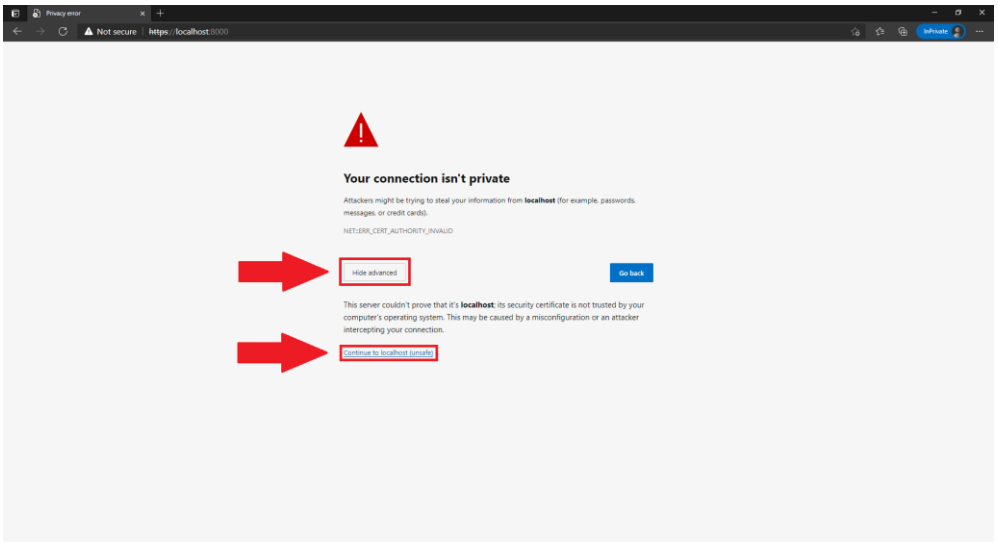
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.escuelaing.arep.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.031 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.468 s
[INFO] Finished at: 2021-03-10T14:13:51-05:00
[INFO] ------------------------------------------------------------------------
```

Pruebas en Localhost

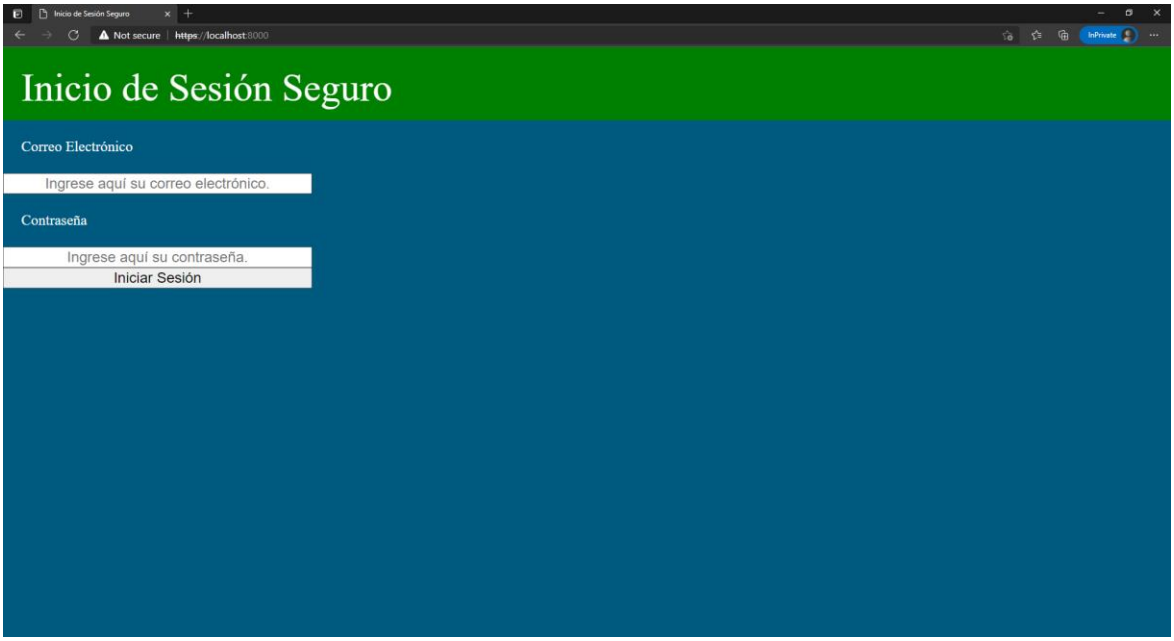
Para verificar que en la aplicación Docker se hayan desplegado con éxito los contenedores LoginSeguro y ServicioSeguro en sus respectivos puertos, se abre la aplicación de Docker de escritorio y se hace la verificación que todos los contenedores estén corriendo en sus respectivos puertos. Como se ve en la siguiente imagen, todos los contenedores están corriendo satisfactoriamente.



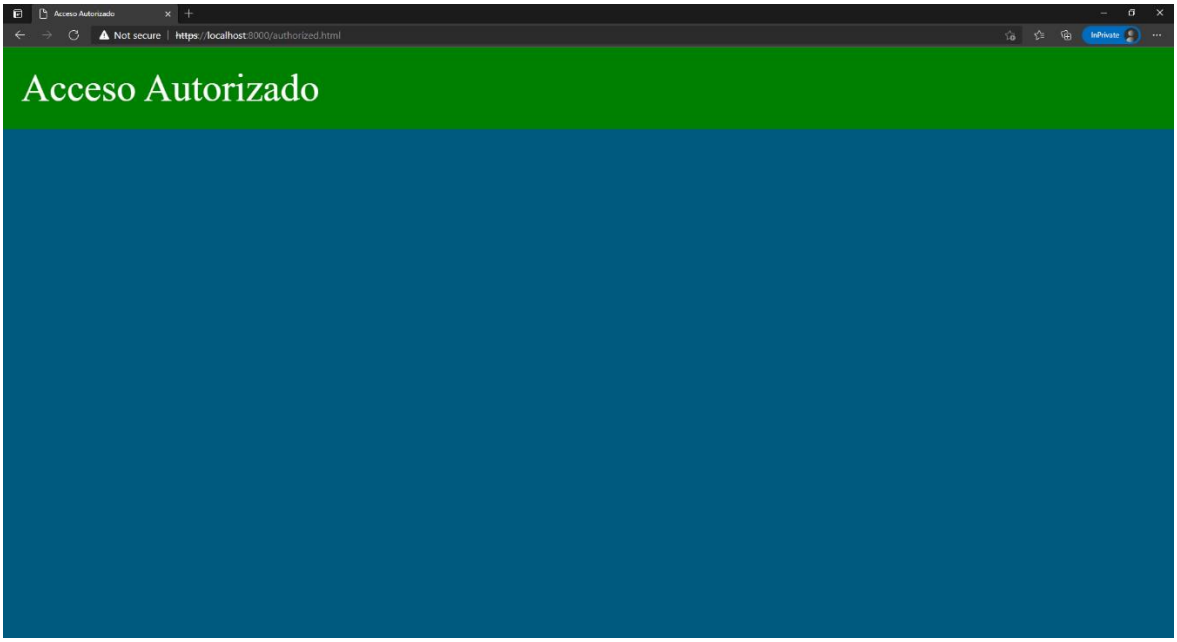
Para comprobar que la página web ha sido desplegada con éxito de manera local utilizando Docker, se ingresa en el navegador la siguiente URL: https://localhost:8000/. Como se puede observar, el contenedor ha sido desplegado satisfactoriamente de manera local utilizando Docker. Para poder acceder a ella, se requiere presionar primero clic en **Advanced** y luego en **Continue to localhost (unsafe)**, como se observa a continuación.



Luego de realizar clic en **Continue to localhost (unsafe)**, se observa que la página ha sido desplegada satisfactoriamente de manera local utilizando Docker.

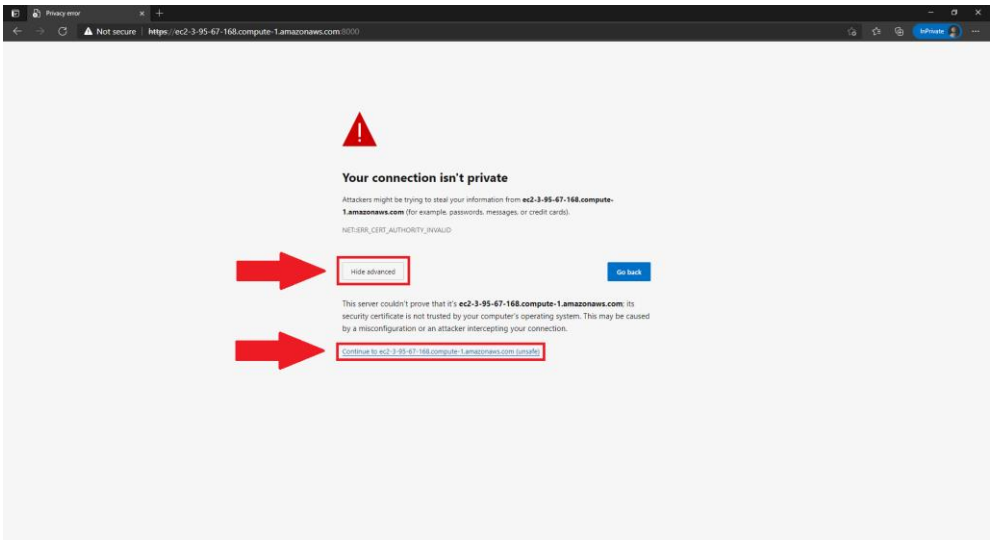


Luego de ingresar las credenciales y presionar clic en el botón Iniciar Sesión, se observa claramente que la página web redirige al recurso https://localhost:8000/authorized.html, lo cual indica que el inicio de sesión ha sido exitoso y seguro, ya que este segundo recurso se maneja también usando el protocolo HTTPS, indicando que la aplicación garantiza autenticación, autorización e integridad de usuarios.

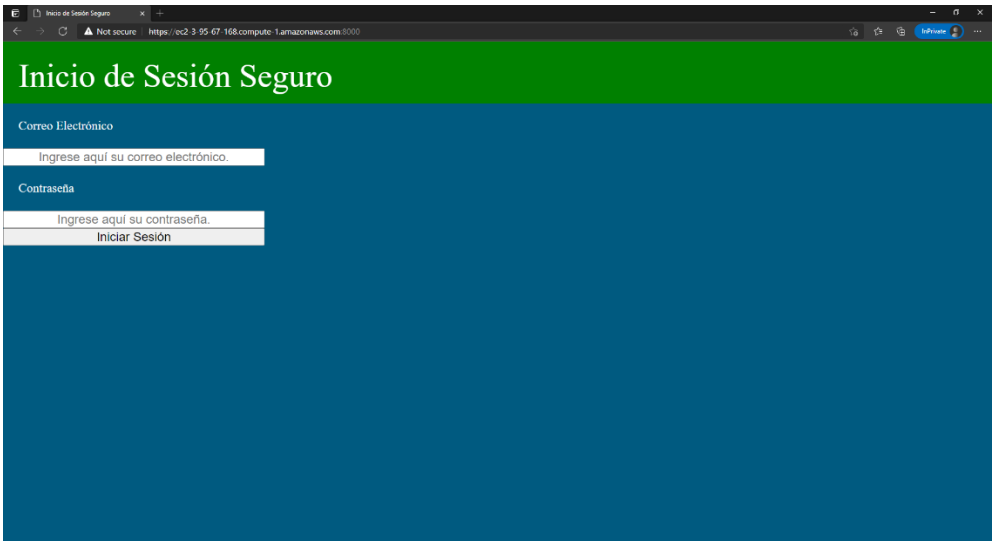


Pruebas en AWS

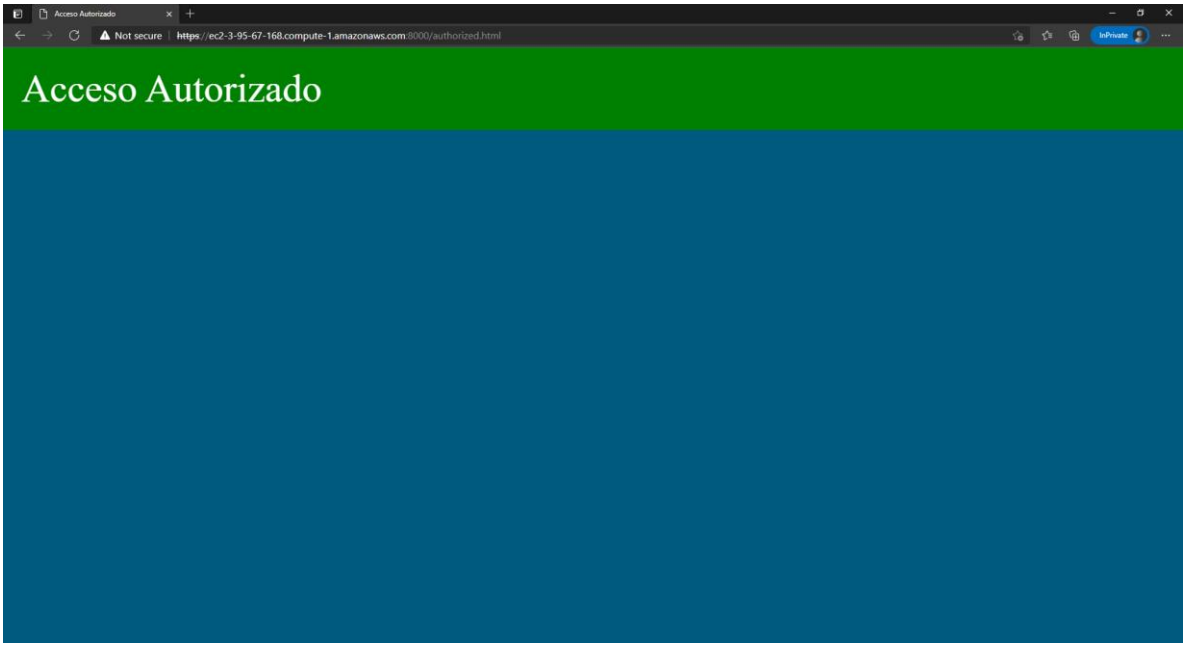
Para realizar las pruebas correspondientes de la ejecución del programa en AWS, y que el contenedor se encuentra activo desde la máquina virtual, ingresamos en el navegador la siguiente URL: https://ec2-3-95-67-168.compute-1.amazonaws.com:8000. Como se puede observar, el contenedor ha sido desplegado satisfactoriamente desde la máquina virtual montada en AWS. Para poder acceder a ella, se requiere presionar primero clic en **Advanced** y luego en **Continue to ec2-3-95-67-168.compute-1.amazonaws.com (unsafe)**, como se observa a continuación.



Luego de realizar clic en **Continue to ec2-3-95-67-168.compute-1.amazonaws.com (unsafe)**, se observa que la página ha sido desplegada satisfactoriamente en la máquina virtual.

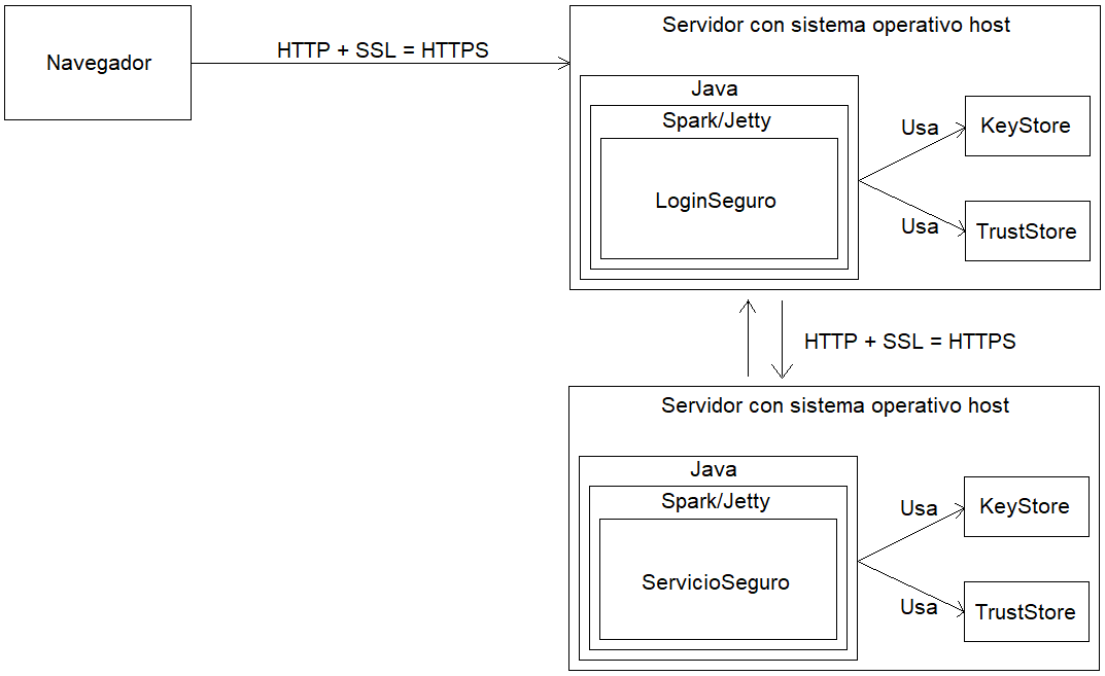


Luego de ingresar las credenciales y presionar clic en el botón **Iniciar Sesión**, se observa claramente que la página web redirige al recurso https://ec2-3-95-67-168.compute-1.amazonaws.com:8000/authorized.html, lo cual indica que el inicio de sesión ha sido exitoso y seguro, ya que este segundo recurso se maneja también usando el protocolo HTTPS, indicando que la aplicación garantiza autenticación, autorización e integridad de usuarios.



5. ARQUITECTURA

- La arquitectura planteada para la realización de todo el programa fue primero planteada para que funcionara de manera local, luego se implementó en el archivo **Docker** todo lo necesario para poder desplegarlo en Docker, y, por último, se realizaron pasos adicionales en AWS para poder también desplegarlo en una máquina virtual en AWS para poder desplegarlo satisfactoriamente.
- Para la implementación del programa **LoginSeguro** se utilizaron dos endpoints, uno de ellos es el login, que es donde el usuario inicia sesión de forma segura utilizando el protocolo HTTPS con una llave y su respectivo certificado. El otro endpoint es information, que es donde se realiza la conexión con **ServicioSeguro**, para así garantizar integridad, autorización y autenticación a nivel de usuario, en el cual se implementó un certificado digital que asegura que se harán consultas de manera confiable y segura en el navegador. También se creó un **myTrustStore** que se encarga de almacenar los certificados de **ServicioSeguro** para tenerlo como una fuente confiable y realizar una comunicación segura.
- Para la implementación de **ServicioSeguro** se utilizó tan solo un endpoint, el cual es information también, que se encarga de retornar el mensaje de **Acceso Autorizado**, para así garantizar integridad, autorización y autenticación. A nivel de servidores se realizó la implementación de un certificado digital, y, asimismo, se realizó un **myTrustStore** el cual se encarga de almacenar el certificado y así poder otorgárselo a **ServicioSeguro** y realizar una comunicación de manera segura y confiable.
  La arquitectura detallada de la implementación de todo el programa se puede describir con el siguiente diagrama.



6. CONCLUSIONES

- En el laboratorio se utilizó un programa con los requerimientos preestablecidos en el enunciado propuesto, el cual es capaz de manejar inicio de sesión seguro utilizando llaves con sus respectivos certificados, en la que se manejaron dos programas **LoginSeguro** y **ServicioSeguro**, encargados de desplegar una interfaz de usuario, el cual era capaz de iniciar sesión con el protcolo HTTPS para disponer de una aplicación web garantizando integridad, autorización y autenticación a nivel de usuario, el cual fue posteriormente desplegado en Docker y AWS.
- Por otro lado, se implementó integración continua en todo el código fuente para llevar un control de calidad del código, para asegurar que el código esté funcionando totalmente sin ningún problema tanto de compilación como de los resultados retornados después de realizar las respectivas operaciones, y así poder desplegar la aplicación sin presentar ningún tipo de errores.
- Para asegurar que el código funcionaba utilizando el protocolo HTTPS, se crearon dos llaves y dos certificados para cada una de estas llaves, uno para cada programa, para así garantizar seguridad al usuario cuando ingresara a la interfaz de inicio de sesión, y luego de iniciar sesión, darle acceso autorizado con otra interfaz también utilizando el protcolo HTTPS, para asegurar comunicación entre los dos programas y las dos páginas web, utilizando siempre los respectivos certificados para así asegurar una navegación totalmente segura.
- Finalmente, luego de implementar todo el código, se pudo realizar el contenedor en Docker del código, el cual fue desplegado en Docker primero para comprobar total funcionalidad en Docker, para así proceder con el despliegue en AWS al instalar Docker dentro de la máquina virtual y así poder desplegar el programa en la nube desde la máquina virtual que provee AWS.

7. REFERENCIAS

- Vargas, Julián. “Aprende Sobre Desarrollo Web.” Aprende Sobre Desarrollo Web | MDN, 23 Junio 2015, developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript.
- Delgadillo, Daniel. “CSS, ¿Qué Es?” *Arume*, 1 Abr. 2019, www.arumeinformatica.es/blog/css/.
- Bernal, Gustavo. “¿Qué Es HTML? Explicación De Los Fundamentos.” Tutoriales Hostinger, 10 Dic. 2020, www.hostinger.es/tutoriales/que-es-html/.
- Sánchez, Sebastián. “¿Qué Es Un Framework? - Wild Code School.” Www.wildcodeschool.com, 13 Junio 2019, www.wildcodeschool.com/es-ES/blog/que-es-un-framework.
- Emaz, Antoine. “¿Qué Es Docker?” *Amazon*, UAP, 25 Nov. 2016, aws.amazon.com/es/docker/.
- Rodríguez, Claudio. “Amazon Web Services (AWS) : ¿Qué Es y Qué Ofrece?” *TIC Portal*, 2 Feb. 2021, www.ticportal.es/temas/cloud-computing/amazon-web-services.
- Pérez, Andrés. “HTTPS.” *¿Qué Es El Protocolo HTTPS y Por Qué Es Tan Importante?*, 12 Aug. 2020, es.ryte.com/wiki/HTTPS.
- Cuesta, Ricardo. “¿Qué Son SSL,TLS y HTTPS?” *¿Qué Son SSL, TLS y HTTPS?*, 2 July 2021, www.websecurity.digicert.com/es/es/security-topics/what-is-ssl-tls-https.