Amazon Gateway y Lambda

Alejandro Toro Daza

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

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

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En el taller Amazon Gateway y Lambda se desarrolló una aplicación capaz de mediante un servicio web en Spark, convertir de grados Fahrenheit a grados Celsius. Este servicio responde con un JSON, el cual fue desplegado en una máquina de AWS EC2, en la que se creó una ruta en API Gateway para así acceder al servicio, en la que la integración como tal no fue con función lambda, sino con la API. Posteriormente se creó una aplicación JS en la que se usa el servicio, la cual fue desplegada en S3, y se aseguró que estuviera disponible en internet desde cualquier máquina, para así comprobar que la aplicación fue desplegada con éxito y que puede ser usada desde cualquier máquina con acceso a internet. Para esto, fue necesario crear primero la aplicación en Java con interfaz de Usuario, en la que el Usuario puede ingresar el valor en Fahrenheit, para posteriormente ver por medio de la API que fue creada luego de crear la función lambda, el resultado de la conversión.

2. OBJETIVOS

- Crear un servicio Web en Spark que convierta de grados farenheit en grados celcius, el cual debe responder un JSON.
- Desplegar el servicio en una máquina de AWS EC2 para posteriormente publicarla.
- Crear una ruta en el API Gateway para acceder al servicio, en la cual la integración no es con función lambda.
- Crear una aplicación JS para usar el servicio y desplegar la aplicación en S3. También asegurarse que esté disponible sobre internet.

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.
- **Función Lambda:** Las expresiones lambda suelen utilizarse para lo siguiente:
  - Como argumentos que son pasados a otras funciones de orden superior.
  - Para construir el resultado de una función de orden superior que necesita retornar una función.

  Si la función lambda sólo se utiliza una vez es utilizada una vez o un número limitado de veces, una expresión lambda puede ser sintácticamente más simple que una función nombrada. Las funciones lambda son muy comunes en la programación funcional y otros lenguajes con funciones de primera clase, donde cumplen el mismo papel para el tipo de función como literales para otros tipos de datos.

  Formas de las expresiones lambda

  Una expresión lambda es una expresión que tiene cualquiera de estas dos formas:

  Lambda de expresión

  Una lambda de expresión que tiene una expresión como cuerpo:
  (input-parameters) => expression
  Una expresión lambda con una expresión en el lado derecho del operador se denomina lambda de expresión. Las lambdas de expresión se usan ampliamente en la construcción de árboles de expresión.

  Lambda de instrucción

  Una lambda de instrucción que tiene un bloque de instrucciones como cuerpo:
  (input-parameters) => { <sequence-of-statements> }
  Como ves, las lambdas de instrucción son similares a las lambdas de expresión, salvo que las instrucciones se encierran entre llaves.

  Inferencia de tipos en expresiones lambda

  A la hora de escribir lambdas, no es necesario especificar un tipo para los parámetros de entrada, ya que el compilador puede deducir el tipo según el cuerpo de lambda, los tipos de parámetros y otros factores. Para la mayoría de los operadores de

consulta estándar, la primera entrada es el tipo de los elementos en la secuencia de origen.

Reglas generales para la inferencia de tipos de las lambdas

La lambda debe contener el mismo número de parámetros que el tipo delegado.

Cada parámetro de entrada de la lambda debe poder convertirse implícitamente a su parámetro de delegado correspondiente.

El valor devuelto de la lambda (si existe) debe poder convertirse implícitamente al tipo de valor devuelto del delegado.

Para saber qué son las expresiones lambda, debemos destacar que no tienen tipo en sí mismas, ya que el sistema de tipos comunes no tiene ningún concepto intrínseco de «expresión lambda».

Lambda en la programación de aplicaciones

Si nos centramos en el ámbito de la programación de aplicaciones, con la adición de expresiones lambda podemos crear códigos de progamación más concisos y significativos, además de abrir la puerta hacia la programación funcional en Java, en donde las funciones lambda juegan un papel fundamental.

Mediante la poderosa combinación de Lambdas y Stream, Java está realizando un cambio de paradigma en su codificación hasta el momento.

Por medio de expresiones lambda podemos referenciar métodos anónimos o métodos sin nombre, lo que nos permite escribir códigos más claros y concisos que cuando usamos clases anónimas.

Una expresión lambda en programación se compone de:

Listado de parámetros separados por comas y encerrados en paréntesis, por ejemplo: (a,b).

El símbolo de flecha hacia la derecha: ->

Un cuerpo que puede ser un bloque de código encerrado entre llaves o una sola expresión.

- **API REST:** El término REST (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones que tener en cuenta en la arquitectura software que usaremos para crear aplicaciones web respetando HTTP.

  Según Fielding las restricciones que definen a un sistema RESTful serían:

  - **Cliente-servidor:** El servidor se encarga de controlar los datos mientras que el cliente se encarga de manejar las interacciones del usuario. Esta restricción mantiene al cliente y al servidor débilmente acoplados (el cliente no necesita conocer los detalles de implementación del servidor y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente).
  - **Sin estado:** aquí decimos que cada petición que recibe el servidor debería ser independiente y contener todo lo necesario para ser procesada.

- **Cacheable:** debe admitir un sistema de almacenamiento en caché. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.
- **Interfaz uniforme:** define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección o "URI".
- **Sistema de capas:** el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

Hoy en día la mayoría de las empresas utilizan API REST para crear servicios web. Esto se debe a que es un estándar lógico y eficiente. Por poner algún ejemplo tenemos los sistemas de identificación de Facebook o también la autenticación en los servicios de Google (hojas de cálculo, Google Analytics, Google Maps, …).

Métodos en una API REST

Las operaciones más importantes que nos permitirán manipular los recursos son:

- **GET** es usado para recuperar un recurso.
- **POST** se usa la mayoría de las veces para crear un nuevo recurso. También puede usarse para enviar datos a un recurso que ya existe para su procesamiento. En este segundo caso, no se crearía ningún recurso nuevo.
- **PUT** es útil para crear o editar un recurso. En el cuerpo de la petición irá la representación completa del recurso. En caso de existir, se reemplaza, de lo contrario se crea el nuevo recurso.
- **PATCH** realiza actualizaciones parciales. En el cuerpo de la petición se incluirán los cambios a realizar en el recurso. Puede ser más eficiente en el uso de la red que PUT ya que no envía el recurso completo.
- **DELETE** se usa para eliminar un recurso.

Otras operaciones menos comunes, pero también destacables son:

- **HEAD** funciona igual que **GET** pero no recupera el recurso. Se usa sobre todo para testear si existe el recurso antes de hacer la petición GET para obtenerlo (un ejemplo de su utilidad sería comprobar si existe un fichero o recurso de gran tamaño y saber la respuesta que obtendríamos de la API REST antes de proceder a la descarga del recurso).
- **OPTIONS** permite al cliente conocer las opciones o requerimientos asociados a un recurso antes de iniciar cualquier petición sobre el mismo.

- **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.

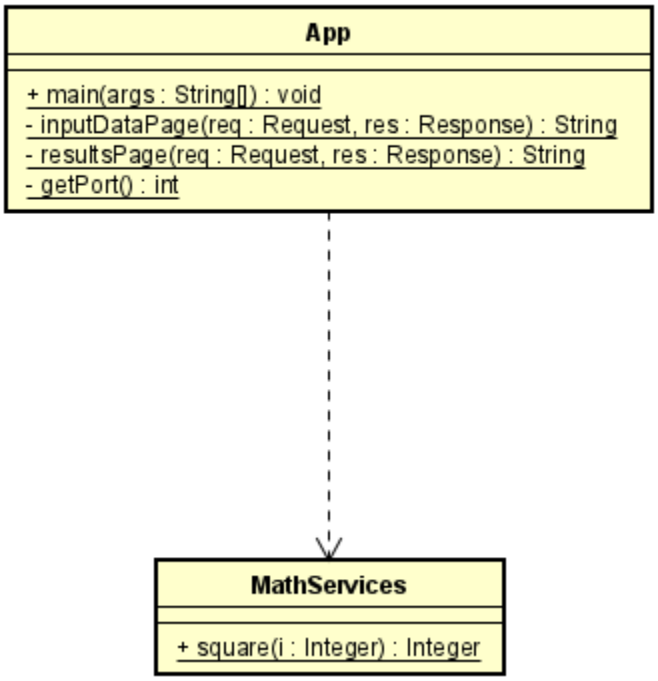
4. DISEÑO Y CONSTRUCCIÓN

Para la creación del proyecto, se dispuso de un solo programa llamado **AmazonGatewayYLambda**. Este programa tiene dos clases encargadas de la ejecución de todo el programa, **App** y **MathServices** respectivamente, y una clase en la que se realizaron las respectivas pruebas, llamada **AppTest**. La clase **App** es la clase encargada de llevar a cabo la ejecución de todo el programa, en la que utilizando el framework de Spark, se creó una interfaz de usuario en formato HTML en el método **inputDataPage**, y en el método **resultsPage** se encarga de redirigir a la página de la API ya implementada desde AWS para retornar el resultado de la conversión de Fahrenheit a Celsius. La clase **MathServices** se encarga de realizar los respectivos cálculos para convertir de grados Fahrenheit a grados Celsius, la cual dispone de un método llamado **square**, que se encarga de realizar las operaciones correspondientes para poder realizar la respectiva conversión de grados Fahrenheit a grados Celsius, utilizando la siguiente fórmula para convertir de grados Fahrenheit a Celsius:

$$C = (F - 32)\frac{5}{9}$$

Teniendo en cuenta que:
C: Celsius.
F: Fahrenheit.

Diagrama de Clases del programa AmazonGatewayYLambda:



4.1. PRUEBAS

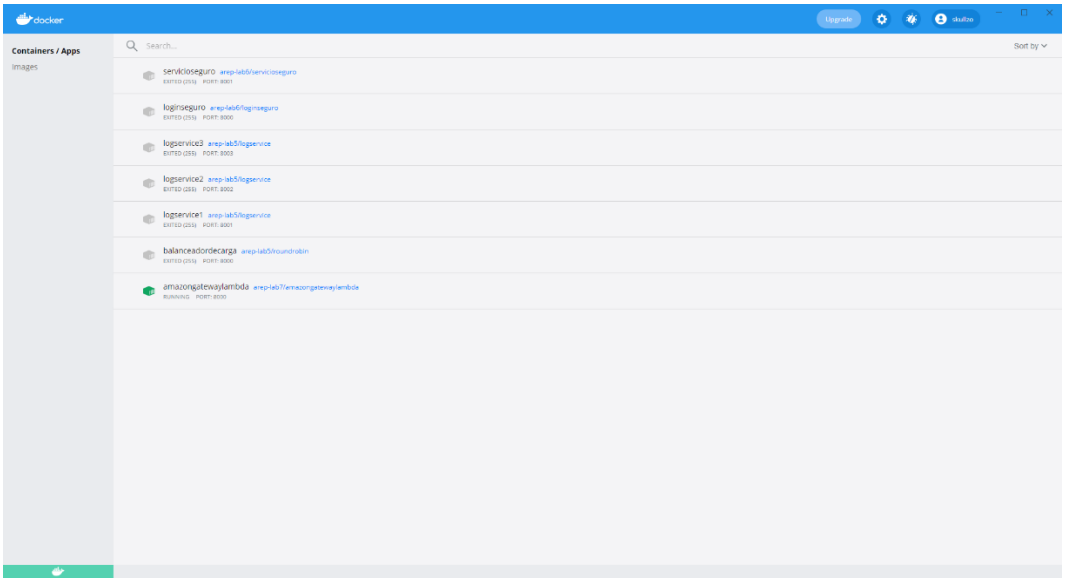
Pruebas en Maven

Para verificar que todo el código está funcionando con total normalidad, se implementaron las pruebas en **Junit**, las cuales fueron implementadas para el programa **AmazonGatewayYLambda**. Para comprobar que las pruebas compilaron correctamente en **AmazonGatewayYLambda**, 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.

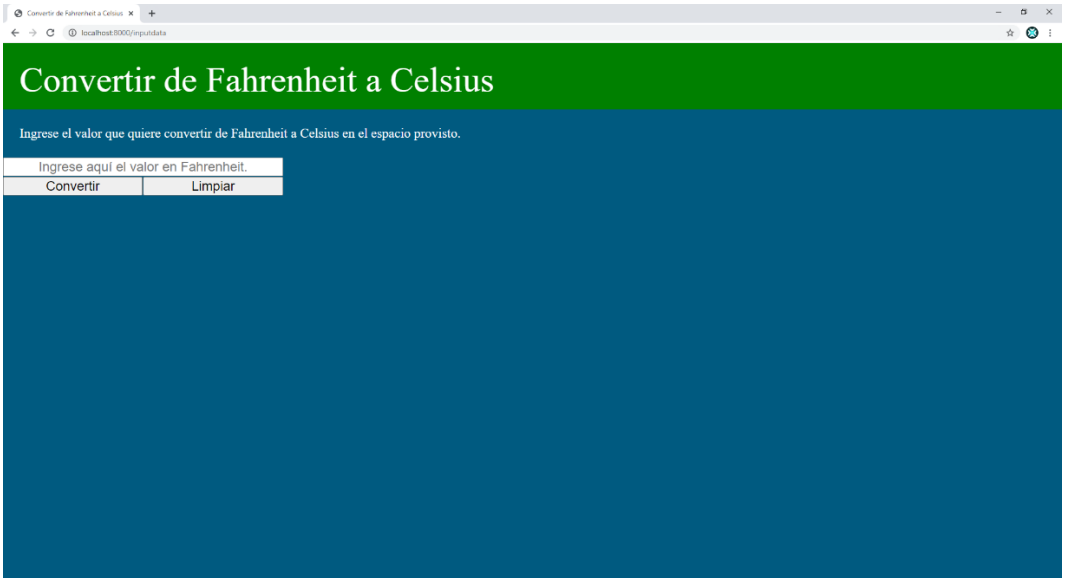
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.escuelaing.arep.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.01 sec

Results :

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

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.282 s
[INFO] Finished at: 2021-03-17T20:47:03-05:00
[INFO] ------------------------------------------------------------------------
```

Pruebas en Docker en Localhost

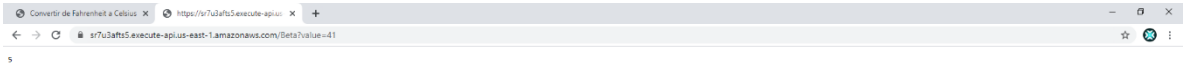
Para verificar que en la aplicación Docker se hayan desplegado con éxito los contenedores en sus respectivos puertos, se abre la aplicación de Docker de escritorio y se hace la verificación que todos los contenedores estén corriendo en sus respectivos puertos. Como se ve en la siguiente imagen, todos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

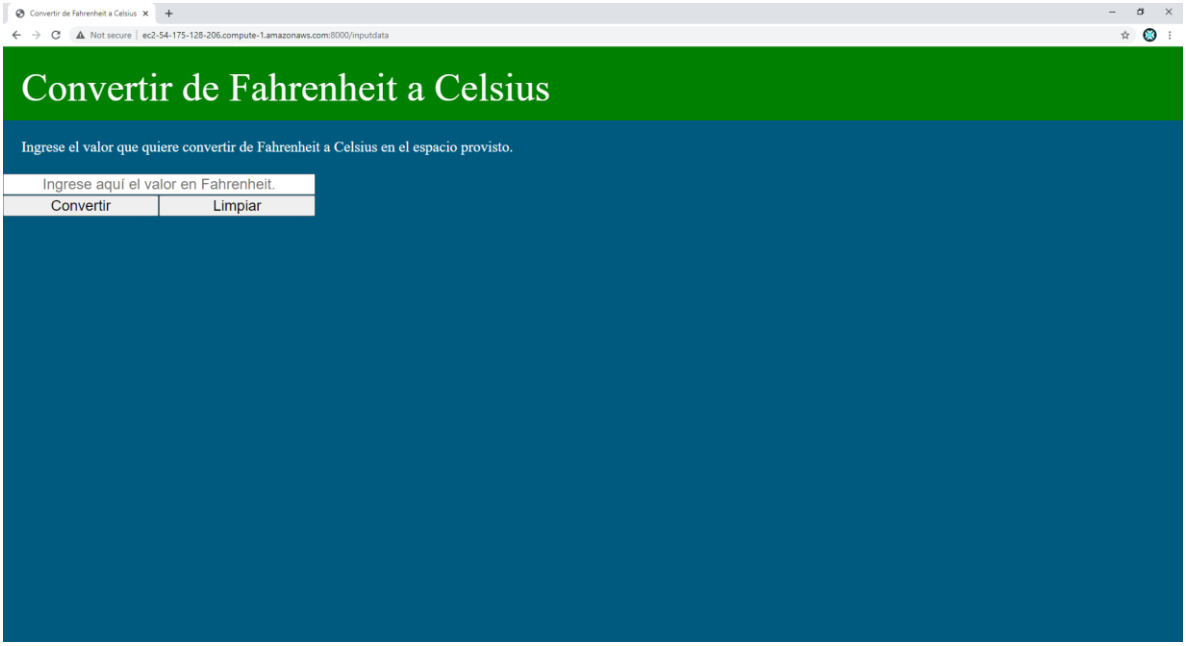


Luego de haber ingresado el valor en Fahrenheit (que para este caso se ha ingresado 41 grados Fahrenheit para realizar la prueba de la página web), se realiza clic en el botón **Convertir**. Luego de realizar clic en el botón **Convertir**, se muestra que se despliega en otra pestaña el resultado, que es **5 grados Celsuis**, y al ver la URL de la nueva pestaña, se ve que es la URL de la API que ha sido creada desde AWS, por lo que el programa está haciendo el uso respectivo de esta API, dando como resultado que el programa está funcionando correctamente.

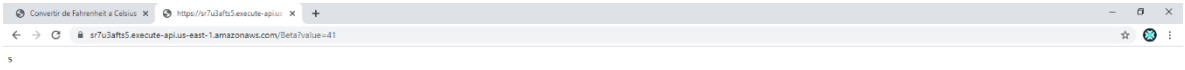


Pruebas en AWS

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: http://ec2-54-175-128-206.compute-1.amazonaws.com:8000/inputdata. Como se puede observar, el contenedor ha sido desplegado satisfactoriamente desde la máquina virtual montada en AWS.



Luego de haber ingresado el valor en Fahrenheit (que para este caso se ha ingresado 41 grados Fahrenheit para realizar la prueba de la página web), se realiza clic en el botón **Convertir**. Luego de realizar clic en el botón **Convertir**, se muestra que se despliega en otra pestaña el resultado, que es **5 grados Celsuis**, y al ver la URL de la nueva pestaña, se ve que es la URL de la API que ha sido creada desde AWS, por lo que el programa está haciendo el uso respectivo de esta API, dando como resultado que el programa está funcionando correctamente.



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 **AmazonGatewayYLambda**, se realizó con el framework Spark la conexión HTTP que maneja el usuario con el servidor, en la cual el Usuario realiza su petición al servidor ingresando a la página, y al ingresar el valor que desea convertir de Fahrenheit a Celsius, dentro de la función lambda, se ejecuta la clase **square** desde el *.jar* que fue utilizado para la función lambda, y por medio de la API REST, se realiza un GET, con el cual de parámetro **value** se ingresa el valor registrado por el usuario. Con ese GET, la API REST resuelve retornando el valor correspondiente después de realizar las respectivas operaciones en el programa **AmazonGatewayYLambda** para realizar la respectiva conversión. El programa **AmazonGatewayYLambda** también tiene una interfaz realizada en HTML, en la cual el usuario dispone de un campo para ingresar el valor en Fahrenheit, y tiene dos botones, uno que es **Convertir** que el usuario usa para realizar la respectiva conversión, y otro que es **Limpiar** que se encarga de quitar los valores ingresados en el espacio provisto, por si el usuario desea ingresar otro valor.

6. CONCLUSIONES

- En el laboratorio se utilizó un programa con los requerimientos preestablecidos en el enunciado propuesto, el cual es capaz de manejar peticiones por parte del usuario, el cual dispone de una interfaz de usuario para poder ingresar el valor en Fahrenheit, para realizar la respectiva conversión a Celsius, utilizando una API REST de la cual se utilizó una función lambda para su respectiva ejecución. Luego de implementar el código, y desplegar la API REST con su respectiva función lambda, el programa 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, primero se probó la función lambda antes de proceder a crear la API REST, para asegurar la calidad y el correcto funcionamiento del código antes de proceder a la creación de la API REST. Al crear la API REST, también antes de desplegarla se realizaron pruebas con una plantilla en formato JSON para comprobar que primero servía ingresando valores enteros, para después agregar el parámetro **value** para así posteriormente probar con cualquier valor, y así asegurar funcionamiento completo del programa.
- Finalmente, luego de implementar todo el código con su respectiva función lambda y API REST, 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.
- Rodríguez, Carlos. “Que Son Las Expresiones Lambda y Su Uso En La Programación.” *Tokio School*, 9 Dic. 2019, www.tokioschool.com/noticias/expresiones-lambda-uso-programacion-aplicaciones/.
- Ordóñez, Jonathan. “¿Qué Es Una API REST?” *Idento*, 9 Mar. 2021, www.idento.es/blog/desarrollo-web/que-es-una-api-rest/.