CDD Cambio de Divisas

By: Luis Fernando Pizza Gamba

Escuela Colombiana de Ingeniería Julio Garavito

Ingeniería de Sistemas

Arquitecturas Empresariales

2019-1

1. Introduction

En este documento se presentará la arquitectura del sistema de CDD (Cabio de divisas) es una aplicación que permite convertir divisas, esta esta desarrollada en un escaparate de diversas tecnologías. Para el desarrollo de esta se buscó implementar toda en html5 utilizando Bootstrap 3 (getbootstrap.com), Javascript, AJAX y JQuery para proporcionar una interfaz de usuario compatible con HTML5 y sensible. Se usaron los servicios de AWS, S3 para alojar y proporcionar un servidor a nuestra aplicación, funciones lambda desarrollada en java 8 sobre un proyecto Maven para La información del tipo de cambio se obtiene a través de una API RESTful. Esta función lambda fue desplegada en un API Gateway y gracias a axios poderla consumir en nuestra aplicación web.

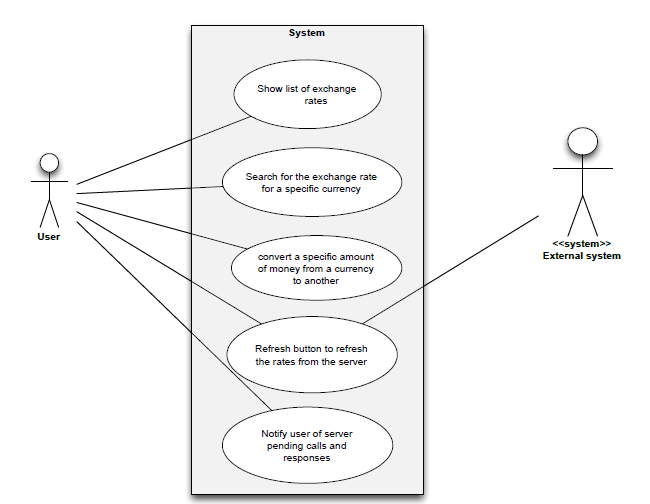
1. Análisis de requerimientos.

Usted es un arquitecto en una organización que proporciona soluciones modernas y fáciles de escalar.

Un cliente del sector financiero desea construir un portal que exponga servicios para el manejo rápido de las transacciones que efectúa la mesa de dinero. El primer conjunto de servicios que se incluirá en la plataforma será el de tasas de cambio. Usted debe diseñar e implementar una aplicación para la consulta y búsqueda de tasas de cambio y para la conversión de monedas. La aplicación debe construirse con microservicios usando la infraestructura de AWS lambda, AWS API Gateway y otros que usted considere. El aplicativo debe tener una interfaz de usuario simple, el servidor es un servidor de publicación de tasas de cambio y el canal de comunicación es TCP o HTTP.

La aplicación podrá extenderse con otros conjuntos de microservicios, es decir su implementación inicial incluirá los requerimientos de tasa de cambio, pero debería poder extenderse a otros servicios financieros, por ejemplo, para negociar bonos, o bitcoins.

El servidor expondría diferentes servicios para el consumo de la aplicación cliente, lista de tasas de cambio para diferentes monedas, búsqueda del tipo de cambio para una moneda específica y un conversor de moneda para convertir una cantidad específica de dinero de una moneda a otra. Utilice una de las API de conversión de moneda pública (https://openexchangerates.org/signup/free,https://currencylayer.com/documentation, etc.) en el lado del servidor para obtener las últimas tasas de conversión.  
  
La aplicación cliente debe proporcionar una interfaz simple y clara para que el usuario final realice diferentes funcionalidades expuestas por el servidor, listando las monedas con sus tasas de cambio, el cuadro de búsqueda de una moneda específica, la caja de cambio de moneda y un botón de actualización para actualizar las tarifas del servidor .  
  
La aplicación cliente debe ser responsive, cualquier congelación en la aplicación cliente no es aceptable, las operaciones de la red deben ser operaciones sin bloqueo, se debe notificar al usuario que hay una llamada del servidor y recibir una notificación una vez que se reciba la respuesta y se presente sin bloqueos.  
  
Debe utilizar tecnologías Java y JS modernas para diseñar e implementar la aplicación con las especificaciones mencionadas.



1. Lista de tipos de cambio para diferentes monedas.

2. Busque el tipo de cambio de una moneda específica

3. Un convertidor de moneda para convertir una cantidad específica de dinero de una moneda a otra.

4. Actualizar botón para actualizar las tasas desde el servidor.

5. Las operaciones de la red deben ser operaciones sin bloqueo, se debe notificar al usuario que hay una llamada del servidor y recibir una notificación una vez que se recibe la respuesta y se procesa sin bloqueos.

4. Responsiva: la aplicación cliente debe ser receptiva, cualquier congelación en la aplicación cliente no es aceptable,

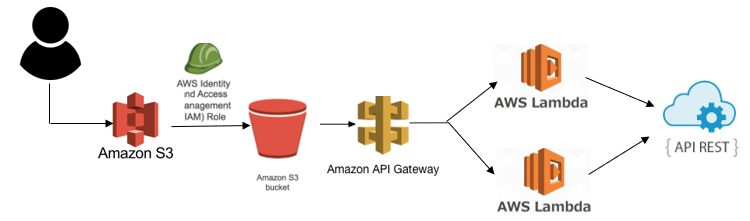
5.Sin bloqueo: las operaciones de red deben ser operaciones sin bloqueo.

6. Usabilidad: la aplicación cliente debe proporcionar una interfaz simple y clara para el usuario final

7. Escalabilidad: La aplicación debe ser escalable.

1. Arquitectura:

La arquitectura de nuestro sistema se basó en 3 capas. A continuación, se encontrará una imagen con la arquitectura y seguido la explicación de cada capa:

****

El usuario accederá por medio del protocolo HTTP al dominio que nos brinda Amazon s3, este buscara dentro de su bucket donde se encuentra nuestro front-end y acedera al recurso html asignado que en este caso es index.html, este como su extensión lo indica esta desarrollado en HTML5 usando como framework boostrap y usando como framework para javaScript Jquery, proporcionándonos una interfaz responsiva. Para los diversos microservicios que ofrece nuestra aplicación estos serán implementados en java script, por medio de AXIOS (cliente HTTP basado en Promesas) consumiremos nuestro API Gateway, axios nos retornara un Json con la información necesaria sobre las divisas. Como capa dos tenemos a nuestro En API Gateway implementaremos nuestras funciones lambda y configuramos que el valor a retornar sea un json. Se tuvo que activar CORS para que desde nuestro s3 con el JavaScript podamos consumir el API Gateway. La primera función lambda consumirá los nombres y siglas de todas las divisas del api rest elegimos. La segunda consumirá los valores de todas las divisas frente al dólar estadounidense USD. Como capa tres tenemos a nuestras funciones lambda están implementadas sobre java 8 en un proyecto Maven, esta se encargará de consumir nuestro api Rest y retornar un json con las diferentes divisas y sus valores respectivos respecto a USD.

Bibliografía:

Diálogos y explicaciones con Andres Florez.