APLICACIÓN WEB PROTOTIPO PARA LA EVALUACIÓN DE COSTOS Y APROVISIONAMIENTO CON LIBCLOUD SOBRE AMAZON WEB SERVICES

PRICECLOUD

INFORME - PRIMERA PARTE

SEBASTIÁN AFANADOR FONTAL
sebastian.afanador@correounivalle.edu.co
Código 1629587

JOHN ALEXANDER SANABRIA ORDÓÑEZ PH.D
john.sanabria@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Universidad del Valle
Cali - Colombia

Agosto 2022 – versión 1.0.0

[4 de agosto de 2022 19:03 – versión 1.0.0]

Las tecnologías usadas en la primera parte del proyecto fueron NodeJS con Sequelize, React, XPath y PostgreSQL.

Docker Compose permitió el despligue sobre Internet de los servicios en un stack de contenedores, el depligue realizado puede encontrarse en la web `https://api.pricecloud.enerfris.com`.

RESUMEN

La aplicación web *Pricecloud* propone integrar la fase de evaluación de costos de los principales proveedores de servicios de Cloud Computing (CC) junto con su despligue en una única interfaz para dar un criterio informado que ayude a minimizar los costos de funcionamiento de un proyecto de software con respecto a sus necesidades iniciales.

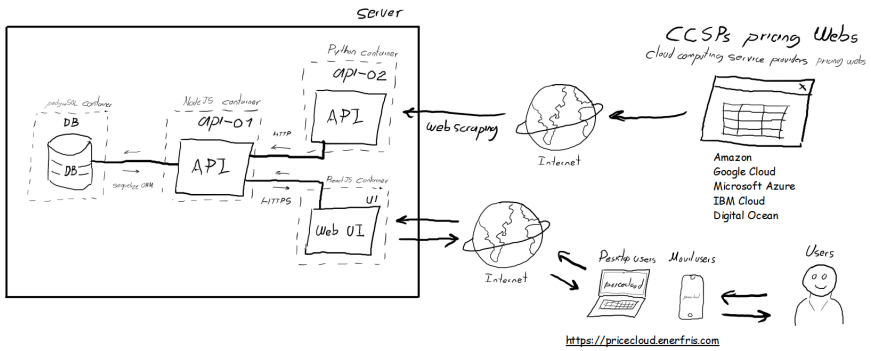
La primera parte de este trabajo tuvo como objetivo principal resolver los objectivos específicos propuestos en el anteproyecto y nombrados a continuación.

- Definir un modelo general de evaluación de costos para el análisis de los servicios en el CC usando información recuperable y relevante de la Web de sus proveedores.
- Diseñar e implementar un prototipo basado en microservicios que recopile periodicamente y almacene las tarifas de los principales CCSPs.

ÍNDICE GENERAL

1 ARQUITECTURA

1.1 ESQUEMA DE SERVICIOS



1.2 DESCRIPCIÓN DE LOS SERVICIOS

La primera parte del proyecto de grado atiende las actividades relacionadas con la etapa de *Creación del modelo de evaluación de costos*f y con las actividades relacionadas con la etapa de *Creación del prototipo de microservicios de backend*. El servicio de backend está compuesto por tres microservicios con funciones específicas cada uno descritas a continuación.

1.2.1 *API REST* api-01

El primer microservicio del backend se encarga de autenticar los usuarios de la aplicación y centralizar todas las peticiones. Este servicio está expuesto en Internet en la dirección `https://api.pricecloud.enerfris.com` se comunica usando el protocolo Hypertext Media Transfer Protocol Secure (HTTPS) con el servicio de la interfaz web autenticando cada petición con *JSON Web Tokens* cifrados con Advanced Encryption Standard (AES); La responsabilidad de este servicio es responder todas las peticiones creadas por el frontend de la aplicación incluyendo las consultas de los precios de los Cloud Computing Service Providers (CCSPs) almacenadas en la base de datos.

1.2.2 *API REST* api-02

El segundo microservicio de backend se encarga de recopilar la información de precios de los CCSPs. Usando técnicas de *Web Scraping* periodicamente es invocado por *api-01* cual se encarga de recibir la

A cada uno de los servicios enunciados en este informe le corresponde una carpeta dentro del repositorio del proyecto. El código fuente solo es accesible por los colaboradores y se encuentra en `https://github.com/sebastianaf/pricecloud`.

La primera parte del proyecto no comprende ningun aspecto de la interfaz gráfica, sin embargo fue necesario implementar el Login *que usará la aplicación para configurar el esquema básico de autenticación.*

Las credenciales del usuario administrador de la API son: ***usuario:*** admin, ***contraseña:*** price22cloud.

información y gestionar su almacenamiento en la base de datos, este microservicio se basa en XPath sobre *Python*.

1.2.3 *Base de datos* db

Finalmente el último microservicio del backend corresponde a la base de datos. En el respositorio del proyecto no existe mas código relacionado con este servicio que el implementado por el *script de despliegue* en el archivo *docker-compose.yml* esto se debe a que todo el esquema de la base de datos reside en *api-01* definido por Sequelize.

1.2.4 *Interfaz Web* ui

Este servicio se encarga de ejecutar el servidor web con la interfaz gráfica de usuario. Implementa el esquema básico de autenticación que se usará en el resto de la aplicación con *api-01* enviando un *token* en las *headers* peticiones HTTPSs.

2 RESUMEN DE ACTIVIDADES

2.1 CRONOGRAMA

A continuación se adjunta información referente al las actividades desarrolladas en la primera parte del trabajo.

Actividad	Meses							
	1	2	3	4	5	6	7	8
Etapa de creación del modelo de evaluación de precios								
Investigar acerca de las los modelos de precios en los CCSP.								
Comparar los criterios de cobro entre cada CCSP.								
Identificar las variables principales que usan los CCSP para el cobro de los servicios.								
Efectuar pruebas de escritorio del modelo para 5 CCSP.								
Etapa de la creación del prototipo de microservicios de backend								
Diseñar el modelo de datos para almacenar la información de costos.								
Diseñar un microservicio usando Crawling y Scraping para recuperar datos de precios de los CCSP.								
Implementar el modelo de evaluación de costos usando un microservicio.								
Diseñar una configuración de microservicios en Docker Compose.								
Desplegar la configuración de microservicios en Docker Compose.								

2.2 1ER OBJETIVO ESPECÍFICO.

Las actividades de este objetivo específico tienen como propósito *definir un modelo general de evaluación de costos para el análisis de los servicios en el CC usando información recuperable y relevante de la Web de sus proveedores.* El resultado obtenido son las ecuaciones del modelo de evaluación de costos listas para implementación. A continuación el resumen de cada actividad.

2.2.1 *Actividades 1 y 2.*

Investigar acerca de las los modelos de precios en los CCSPs y comparar los criterios de cobro entre cada CCSP:

Como resultado de esta actividad y teniendo en cuenta las *variables*

de costo de cada tipo de tipo de servicio en la *sección 2.2.2* se formuló el siguiente criterio.

Dado un servicio bajo demanda s que tienen en común una pareja $ccsp$ donde $ccsp_i$ es un CCSP con $i \in [1, 2]$ se define que:

2.2.1.1 Compute *o* Container

Si s es del tipo *Compute* o *Container* la *variable de costo* principal es la $demanda$ definida como el tiempo en que se usará el servicio. Si se conocen los *rangos de costo* para las variables cpu y ram entonces la *función de costo* del servicio es:

$$costo_i(s, ccsp_i, demanda, cpu, ram) = (costoCPU + costoRAM) * demanda$$

Donde:

- $ccsp_i$: Se define como uno de los Cloud Computing Service Providers (CCSPs) comparados.
- $demanda$: Es el tiempo en horas que se requiere el servicio.
- cpu : Es la cantidad de núcleos de procesamiento requeridos.
- ram : Es la cantidad de memoria RAM requerida en unidades de GB.
- $costoCPU(s, ccsp_i, cpu)$ Es el costo de la cantidad de cpu según el *rango de costo* del $ccsp_i$.
- $costoRAM(s, ccsp_i, ram)$ Es el costo de la cantidad de ram según el *rango de costo* del $ccsp_i$.

El puntaje comparativo $score$ que tiene el $ccsp_1$ con respecto al $ccsp_2$ es:

$$score = \frac{costo_1(s, ccsp_1, demanda, cpu, ram)}{costo_2(s, ccsp_2, demanda, cpu, ram)}$$

Finalmente, si $score < 1$ entonces se dice que el $ccsp_1$ tiene preferencia sobre $ccsp_2$ para el servicio s según un requerimiento de cpu, ram y $demanda$.

2.2.1.2 Storage *o* SQL

Si s es del tipo *Storage* o *SQL* la *variable de costo* principal es la $cuota$ de almacenamiento. Si se conocen los *rangos de costo* para la variable de disponibilidad del recurso llamada $disp$ entonces la *función de costo* del servicio es:

$$costo_i(s, ccsp_i, cuota, disp) = costoDisponibilidad * cuota$$

Donde:

- $ccsp_i$: Se define como uno de los Cloud Computing Service Providers (CCSPs) comparados.
- $cuota$: Es el espacio en GB que se requiere.
- $disp$: Es el tipo de disponibilidad del recurso (Ej: Modo frio, alta velocidad).
- $costoDisponibilidad(s, ccsp_i, cuota, disp)$ Es el costo del tamaño de la $cuota$ para una disponibilidad específica según el *rango de costo* del $ccsp_i$.

2.2.2 *Actividad 3.*

Identificar las variables principales que usan los CCSPs para el cobro de los servicios:

Como conclusión y resultado de esta actividad se definieron las variables principales que usan los CCSPs para el cobro de cada tipo de servicio. Cada variable depende del servicio en cuestión.

En el caso de el servicios tipo *Compute* y *Container* relacionados con máquinas virtuales y contenedores, la variable de costo principal es el *demanda temporal* del recurso en producto con la *cantidad de CPU's* y la *cantidad de memoria RAM*.

Si se habla de un recurso tipo *Storage* y *SQL* la variable de costo principal es la *cuota de almacenamiento* en producto con la *disponibilidad*, este último se relaciona con lo que será almacenado y cómo será recuperado. (Ej: Archivos, objetos o bases de datos. Modo frio o alta velocidad).

2.2.3 *Actividad 4.*

Efectuar pruebas de escritorio del modelo para 5 CCSPs:

Como resultado de este ejercició se concluye que el modelo definido en la *Actividad 1 y 2* puede aplicarse para los servicios básicos de la mayoría de los CCSPs.

2.3 2DO OBJETIVO ESPECÍFICO.

Las actividades de este objetivo específico tienen como propósito *diseñar e implementar un prototipo basado en microservicios que recopile periodicamente y almacene las tarifas de los principales CCSPs*. El resultado

obtenido es un Web service que desempeña las tareas mencionadas. A continuación el resumen de cada actividad.

2.3.1 *Actividad 5.*

Diseñar el modelo de datos para almacenar la información de costos:

Como resultado de esta actividad se diseñó el esquema de la base de datos usando el ORM Sequelize sobre PostgreSQL, Los archivos relacionados con cada Entidad de la base de datos se encuentran en `https://github.com/sebastianaf/pricecloud/tree/master/api-01/src/db/models`.

2.3.2 *Actividad 6.*

Diseñar un microservicio que use técnicas de Crawling *y* Scraping *para recuperar datos de precios de los CCSPs*:

Como resultado de esta actividad se costruyó el servicio escrito en *Python* ubicado en `https://github.com/sebastianaf/pricecloud/tree/master/api-02`. A la fecha de elaboración de este informe se continúa trabajando en recuperar la información de costos en algunos CCSPs mencionados en la *sección 1.1*

2.3.3 *Actividad 7.*

Implementar el modelo de evaluación de costos usando un microservicio:

Como resultado de esta actividad se creó el endpoint ubicado en `https://api.pricecloud.enerfris.com/evaluate`. El código de este endpoint se encuentra en `https://github.com/sebastianaf/pricecloud/tree/master/api-01/src/routes`

2.3.4 *Actividad 8.*

Diseñar una configuración de microservicios en Docker Compose:

Como resultado de esta actividad se codificó el archivo *docker-compose.yml* ubicado en `https://github.com/sebastianaf/pricecloud/blob/master/docker-compose.yml` donde están configurados todos los servicios de la primera parte de la aplicación.

2.3.5 *Actividad 9.*

Desplegar la configuración de microservicios en Docker Compose:

Como resultado de esta actividad se desplegaron dos servicios API REST y un servidor web, la descripcción de estos se encuentra en la sección *sección 1.2* y su código fuente en `https://github.com/sebastianaf/pricecloud`.

Los servicios se encuentran desplegados en las siguientes ubicaciones:

- `https://pricecloud.enerfris.com`
- `https://api.pricecloud.enerfris.com`
- `https://db.pricecloud.enerfris.com`

GLOSARIO

Cloud Computing Service Provider Empresa que presta servicios de Computación en la nube. 1, 4, 5, 9

Cloud Computing El CC es una tendencia de servicios y recursos que funcionan sobre Internet para facilitar el despliegue, uso y disponibilidad de aplicaciones de Software. ii, 9

Data Base Manage System Es un software especializado para administrar bases de datos.. 8, 9

Docker Es una plataforma de contenerización para la implementación microservicios, abstrae recursos como redes, volumenes de almacenamiento, y nodos de cómputo. 8

Docker Compose Es una tecnología para la automatización y orquestación de aplicaciones a partir de contenedores de Docker. ii, 6, 7

Hypertext Media Transfer Protocol Secure El protocolo que usa la web para enviar y recibir tráfico encriptado a través de una capa de sockets seguros. 1, 9

NodeJS Es un entorno de ejecución para Javascript del lado del servidor, está creado para extender el uso del lenguaje fuera del navegador.. ii

PostgreSQL Es un Data Base Manage System (DBMS) muy conocido por ser de código libre y tener extensa compatibilidad con múltiples arquitecturas de hardware.. ii, 6

React Es una biblioteca de Javascript para la construcción de aplicaciones web reactivas. ii

Representational State Transfer API Es una interfaz de programación de aplicaciones que permite la comunicación a través de la web usando el protocolo HTTP. 9

Sequelize Definido como un Object Relational Mapping (ORM) permite definir el modelo de la base de datos usando el lenguaje del backend y simplifica algunas labores de consultas y compatibilidad entre servidores de bases de datos.. ii, 2, 6

XPath Es un lenguaje de programación para formular patrones que permiten encontrar coincidencias en documentos de marcado de etiquetas. ii, 2

SIGLAS

AES Advanced Encryption Standard. 1
API Application Programing Interface. 1
API REST Representational State Transfer API. iii, 1, 7

CC Cloud Computing. ii, 3
CCSP Cloud Computing Service Provider. ii, 1, 3, 4, 5, 6

DBMS Data Base Manage System. 8

HTTPS Hypertext Media Transfer Protocol Secure. 1, 2

ORM Object Relational Mapping. 6, 8