### Nivel 4

#### Objetivos:

* Evolucionar la arquitectura existente para incluir un micro servicio que proporcione el API [POST] /quote, según lo especificado en el enunciado en [Servicio de consulta de condiciones de venta](https://opobla.github.io/nicaventas/Servicio-de-consulta-de-condiciones-de-venta)
* Añadir al archivo docker-compose el nuevo microservicio
* Añadir políticas de cacheo de forma que si se solicita [POST] /quote con los mismos parámetros se responda desde la cache de REDIS en lugar de volver a realizar la consultas a OpenWeather y la BBDD. La valided de uno de estos datos cacheados será de 5 min. Con objeto de verificar que la cache funciona, incluir en la respuesta un campo cache como se hizo anteriormente.

#### Entregables:

* Documento explicativo del trabajo realizado
* Archivo docker-compose
* Script para hacer la carga inicial de los valores iniciales de las tablas de la base de datos.

#### Pruebas a superar:

1) Arrancar los servicios haciendo:

docker-compose up

Hacer pruebas específicas con distintas llamadas para comprobar que se disparan las reglas apropiadas en casa caso y que se cachean las respuestas.

### Se modifico el archivo nivel4.py de la siguiente manera:

### from flask import Flask, jsonify, request, Response

### import json

### import mysql.connector

### #from flask\_cors import CORS, cross\_origin

### import redis

### import pickle

### import hashlib

### import requests

### app = Flask(\_\_name\_\_)

### R\_SERVER = redis.Redis(host='redis',port=6379)

### def Connection():

### return mysql.connector.connect(user='root', host='nivel4-db', port='3306', password='toor', database='nivel4-db-mysql')

### @app.route('/')

### def nivel4():

### return print("Bienvenido al nivel 4")

### @app.route('/active', methods=['GET', 'POST', 'PUT'])

### def active():

### if request.method == 'GET':

### country = request.args.get("country", default = "no country", type = str)

### city = request.args.get("city", default = "no city", type = str)

### key = country+city

### info = {}

### sql\_select = "select city.active, city.city, country.country from city inner join country on country.id = city.id\_country where country.country LIKE '%{}%' and city.city LIKE '%{}%'".format(country, city)

### if R\_SERVER.get(key):

### info = json.loads(R\_SERVER.get(key))

### else:

### try:

### db = Connection()

### cursor = db.cursor()

### cursor.execute(sql\_select)

### records = cursor.fetchall()

### if records == None:

### info = {"Respuesta":"No hay datos o no se pasaron argumentos"}

### else:

### for row in records:

### if row[0] == 0:

### v\_active = False

### else:

### v\_active = True

### info = {"active":"{}".format(v\_active),"country":"{}".format(row[2]),"city":"{}".format(row[1]),"cache":"miss"}

### info\_cache = {"active":"{}".format(v\_active),"country":"{}".format(row[2]),"city":"{}".format(row[1]),"cache":"hit"}

### R\_SERVER.set(key,json.dumps(info\_cache))

### except Exception as e:

### print("Error en SQL:\n",e)

### finally:

### db.close()

### return jsonify(info)

### if request.method == 'POST':

### req\_data = request.get\_json()

### country = req\_data['country']

### city = req\_data['city']

### db = Connection();

### try:

### sql\_insert = "INSERT INTO `country` (`country`) VALUES ('{}')".format(country)

### cursor = db.cursor()

### cursor.execute(sql\_insert)

### db.commit()

### sql\_select = "select id from country where country = '{}'".format(country)

### cursor.execute(sql\_select)

### records = cursor.fetchall()

### for row in records:

### id\_v = row[0]

### sql\_insert\_c = "INSERT INTO `city` (`city`, `id\_country`, `active`) VALUES ('{}', '{}', '1')".format(city, id\_v)

### cursor.execute(sql\_insert\_c)

### db.commit()

### info = {"Response": "Datos guardados correctamente"}

### except Exception as e:

### print("Error in SQL:\n", e)

### finally:

### db.close()

### return jsonify(info)

### if request.method == 'PUT':

### req\_data = request.get\_json()

### country = req\_data['country']

### city = req\_data['city']

### db = Connection();

### try:

### sql\_select = "select city.id,city.active from city inner join country on country.id = city.id\_country where country.country like '%{}%' and city.city like '%{}%'".format(country, city)

### cursor = db.cursor()

### cursor.execute(sql\_select)

### records = cursor.fetchall()

### for row in records:

### city\_id = row[0]

### active = row[1]

### if active == 0:

### active\_n = 1

### info = {"Response": "Se activo la venta"}

### else:

### active\_n = 0

### info = {"Response": "Se desactivo la venta"}

### sql\_update = "update city set active = {} where city.id = {}".format(active\_n, city\_id)

### cursor.execute(sql\_update)

### db.commit()

### except Exception as e:

### print("Error in SQL:\n", e)

### finally:

### db.close()

### return jsonify(info)

### @app.route('/quote',methods=['POST'])

### def quote():

### req\_data = request.get\_json()

### sku = req\_data['sku']

### country = req\_data['country']

### city = req\_data['city']

### key = sku+country+city

### descripcion = ""

### base\_price = 0

### variation = 0

### info = {}

### if R\_SERVER.get(key):

### info = json.loads(R\_SERVER.get(key))

### return jsonify(info)

### else:

### uri = ("http://api.openweathermap.org/data/2.5/weather?q={},{}&appid=3225ae99d4c4cb46be4a2be004226918".format(city,country))

### try:

### uResponse = requests.get(uri)

### except requests.ConnectionError:

### return "Error en la coneccion de la api openweather"

### Jresponse = uResponse.text

### data = json.loads(Jresponse)

### id\_weather = data['weather'][0]['id']

### 

### try:

### sql\_select = "select productos.descripcion, productos.precio from productos where sku = '{}'".format(sku)

### db = Connection()

### cursor = db.cursor()

### cursor.execute(sql\_select)

### records = cursor.fetchall()

### for row in records:

### descripcion = row[0]

### base\_price = row[1]

### except Exception as e:

### print("Error en SQL:\n",e)

### finally:

### db.close()

### 

### try:

### sql\_select\_varia = "select max(variation) from reglas where ciudad = '{}' and pais = '{}' and min\_condition <= {} and max\_condition >= {} and sku = '{}'".format(city,country,id\_weather,sku)

### db = Connection()

### cursor = db.cursor()

### cursor.execute(sql\_select\_varia)

### records = cursor.fetchall()

### for row in records:

### variation = row[0]

### except Exception as e:

### print("Error en SQL:\n",e)

### finally:

### db.close()

### if variation == 0:

### precio\_final = base\_price \* 1

### else:

### precio\_final = base\_price \* variation

### info = {

### "sku":"{}".format(sku),

### "description":"{}".format(descripcion),

### "country":"{}".format(country),

### "city":"{}".format(city),

### "base\_price":"{}".format(base\_price),

### "variation":"{}".format(variation),

### "final\_price":"{}".format(precio\_final),

### "cache":"miss"

### }

### info\_cache = {

### "sku":"{}".format(sku),

### "description":"{}".format(descripcion),

### "country":"{}".format(country),

### "city":"{}".format(city),

### "base\_price":"{}".format(base\_price),

### "variation":"{}".format(variation),

### "final\_price":"{}".format(precio\_final),

### "cache":"hit"

### }

### R\_SERVER.set(key,json.dumps(info\_cache))

### return jsonify(info)

### if \_\_name\_\_ == '\_\_main\_\_':

### app.run(debug=True, host='0.0.0.0', port='8000')

### Se agrego un endpoint /quote en el cual se pasa por POST un objeto JSON:

### La URL a visitar por POST es: <http://localhost:8000/quote>

### Y pasamos un objeto json en el body: por ejemplo

### {

### "sku": "AZ00001",

### "country": "ni",

### "city": "Leon"

### }

### Con estos valores obtendremos una respuesta de un objeto json de la siguiente forma:

### {

### "base\_price": "10.00",

### "cache": "hit",

### "city": "Leon",

### "country": "ni",

### "description": "Paraguas de seÃ±ora estampado",

### "final\_price": "10.00",

### "sku": "AZ00001",

### "variation": "15.00"

### }

### El objeto Json devuelve el precio base del producto, si esta en cache y la variacion que ha tenido.

### Para correr este nivel 4 ponerse en carpeta raiz y ejecutar el siguiente comando:

### **Docker-compose -f nivel4.yml**

### Y se crearan todos los servicios correspondientes.