### Nivel 3

#### Objetivos:

* Modificar el microservicio para que la respuesta de la llamada [GET] /active se haga desde una cache REDIS. La respuesta de este servicio ahora incluirá un campo cache que valdrá miss si la respuesta procede de la base de datos, y hit si la respuesta procede de caché.
* Modificar el microservicio para que la respuesta de la llamada [POST] /active invalide el posible contenido cacheado en REDIS.
* Añadir al archivo docker-compose el servicio de redis

#### Entregables:

* Documento explicativo del trabajo realizado
* Archivo docker-compose

#### Pruebas a superar:

1) Arrancar los servicios haciendo:

docker-compose up

2) Hacer una petición POST con una ciudad, pais y estado concretos. 3) Hacer una petición GET para la misma ciudad y pais. Verificar que para esta primera petición el campo cache deber ser miss. 4) Hacer una segunda petición GET para la misma ciudad y pais. Verificar que para esta segunda petición y las consecutivas el campo cache deber ser hit.

Se agrego en el archivo nivel3.yml la imagen redis para utilizar el servicio de cache con FLASK

El archivo nivel3.py fue modificado para que este utilice redis

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

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

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

def Connection():

return mysql.connector.connect(user='root', host='nivel3-db', port='3306', password='toor', database='nivel3-db-mysql')

@app.route('/')

def nivel3():

return print("Bienvenido al nivel 3")

@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)

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

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

Como pudimos notar ahora el metodo GET de la ruta localhost:8000/active devolvera un JSON que contendra un forma de la siguiente manera:

{

"active": true,

"country": "ni",

"city": "Leon",

"cache": "miss"

}

La clave “cache” tendra el valor de miss o de hit, miss cuando la peticion GET devuelta los datos desde la base de datos y hit cuando esta sea devuelta desde la cache de redis.

Para correr este Nivel3 ubicarse en la carpeta raiz Nivel3 y ejecutar el comando:

docker-compose -f nivel3.yml up

Con este comando se cocinaran todos los contenedores para que a traves de la URL:

<http://localhost:8000/active>

con parametro city=leon&country=ni

Podemos utilizar curl de la siguiente forma:

curl “<http://localhost:8000/active?country=ni&city=leon>”