UNIVERSIDAD TECNOLÓGICA METROPOLITANA
FACULTAD DE INGENIERÍA
Escuela de Informática

Computación Paralela y Distribuida

Proyecto REST

Integrantes:

- Javier López
- Alex Bidart
- Shu-yi Wong

Profesor: Sebastián Salazar
Fecha: 4 de Diciembre de 2020

Índice

Introducción

El presente informe muestra cómo el proyecto REST fue llevado a cabo, mostrando distintos resultados. Los métodos utilizados fueron los de HTTP para REST más ocupados: GET y POST.

Los datos que consumen el servicio REST provienen de una base de datos previamente creada y llenada en servicio MySQL, y los datos a entregar corresponden a la lectura de los datos de todos los países, luego la entrega de la información de un país en específico, también la lectura de un indicador en un país en específico en un año en específico y finalmente existe una ruta con procedimiento POST que entrega los valores de un indicador y un país especificado en el cuerpo entre las fechas correspondientes.

Problemática

“Actualmente la inestabilidad económica, político y social de la región, ha motivado a varias personas a migrar de su país de origen. Sin embargo, la migración es un proceso muy complicado y requiere mucha información antes de tomar una decisión, por este motivo existe una oportunidad para brindar una solución tecnológica que sea un aporte a este nicho de mercado, entregando una aplicación de calidad. El siguiente proyecto requiere que se entreguen un conjunto de indicadores (por cada estados miembro de la Organización de las Naciones Unidas, anual y con un historial de al menos 15 años, más es mejor) que permita hacer comparaciones para apoyar decisiones difíciles.”

Frente al caso anterior citado, se solicita la creación de un servicio REST que consuma los datos proveniente del Banco Mundial, Fondo Monetario Internacional y la Fundación Heritage de manera que puedan modelarse en una base de datos OpenSource, estos datos se consumen por el servicio bajo la premisa de 4 operaciones o rutas.

- Entrega la información de un país en específico.

  /api/countries/{code}/info

- Entrega la información de todos los países

  /api/countries/all

- Entrega la información de un indicador solicitado de un país en específico en un año correspondiente.

  /api/indicators/{countryCode}/{indicatorCode}/{year}/info

- Recibe los datos de un país, el indicador y los años de inicio y final, entregando como resultado los datos del país con el indicador solicitado entre las fechas correspondientes.

  /api/indicators/info

Cada una de las rutas especifican su salida correspondiente, además del código de salida exitoso en código HTTP 200, y en caso de error en codigo HTTP 4XX, cada una de las rutas tiene un método específico de consumo el cual corresponde a GET en las primeras 4 y la última el POST, en caso de errores, el servicio entregará un Json explicativo.

Resolución de la Problemática

En primera instancia se hizo la debida extracción de datos hacia la base de datos, en este caso llamada “database rest.sql” en servicio MySQL, creando diferente tablas para cada indicador, de manera que la lectura de los datos solicitados fuera lo más rápida y entendible posible, el llenado de esto se realizó descargando datos provenientes de las diferentes fuentes recomendadas y posterior a eso con un ETL (Talent Open Studio), se realizó el llenado de cada una de las tablas.

Las tablas de cada indicador contiene los atributos de código (código país alpha3), nombre (nombre del país), indicador (valor correspondiente) y año (año correspondiente), esto con la finalidad de poder responder los llamados GET correspondientes a las rutas referente a los indicadores.

Además se creó una tabla llamada info, la cual contiene atributos de alpha2 (código país alpha 2 según ISO-3166-1), alpha3 (código país alpha3 según ISO-3166-1), codigo_pais_moneda (codigo del pais segun ISO-4217), codigo_moneda (código de la moneda de 3 dígitos según ISO-4217), codigo_menor (código de moneda según ISO-4217), moneda (nombre de la moneda), lengua (idioma del país), nombre (nombre del país), esto con la finalidad de responder y englobar de manera más simplificada los GET correspondiente a la información de un país en específico.

Posterior a esto, se realizó el armado de la API REST, la cual está dividido en “X” actividades.

En primer lugar, se definen e importan todas las librerías necesarias para su realización, las cuales se encuentran descritas en el punto de “Tecnologia usada”, luego se hacen dos configuraciones sumamente necesarias, las cuales corresponden a la definición de la api como instancia de flask.

```
app = Flask(__name__)
```

y la configuración de la lectura de base de datos a utilizar.

```
mysql = MySQL()
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = '123'
app.config['MYSQL_DATABASE_DB'] = 'rest'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)
```

Cabe destacar que para poder usar la base de datos, es necesario que esta se encuentre activa en el mysql, en caso de ubuntu 20.04, se puede instalar mysql-server con pip, luego crear el usuario root con la contraseña 123, luego crear la base de datos llamada rest y posterior a eso, ingresar la base de datos ya creada a la base de datos vacía creada.

1. Entrar al entorno mysql en la terminal con el usuario: Root y contraseña: 123

```
mysql -u root -p
```

2. Crear una base de datos llamada "rest" y luego de creada salir con "exit"

```
CREATE DATABASE rest;
```

3. Agregar la base de datos .sql adjunta a la base de datos en el mysql

```
mysql -u root -p rest < database rest.sql
```

Se crearon usuarios y contraseñas con el fin de generar autorización a la API, los cuales son:

Usuario	Contraseña
usuario1	123
usuario2	456
usuario3	789

bajo la premicioa que se verifique estos antes de ingresar a las rutas, en caso de no ser posible, entrega el error de no autorización

```python
@auth.verify_password
def verify_password(username, password):
    if username in users and check_password_hash(users.get(username), password):
        return username
    else:
        abort(401)
```

Luego se encuentran las rutas correspondientes a los errores y sus entregables en caso que sea necesario utilizarlo, cada uno de errores, 400, 404, 405, 401, 403, 500, con su mensaje .json correspondiente.

```python
@app.errorhandler(400)
def bad_request(error=None):
    message = {
        'ok': False,
        'date' : date,
        'message': 'Petición invalida'
    }
    resp = jsonify(message)
    resp.status_code = 400
    return resp
```

Posterior a esto se crean las diferentes rutas que ayudarán a consumir el servicio.

1) /api/countries/all

Mediante el método GET, la API se conecta a la base de datos y ejecuta la query de *"SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as currencyCode, moneda as currencyName, lengua as lang, nombre as name FROM info"* la cual retorna todos los datos exigidos para esta instancia, luego guarda los resultados para ser transformados en formato json y los entrega.

```python
@app.route('/api/countries/all', methods=['GET'])
def all():
    if (request.method != 'GET'):
        return bad_request()
    else:
        try:
            conn = mysql.connect()
            cursor = conn.cursor(pymysql.cursors.DictCursor)
            cursor.execute("SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as cu
            country = cursor.fetchall()
            respone = jsonify(country)
            respone.status_code = 200
            return respone
        except Exception as e:
            print(e)
        finally:
            cursor.close()
            conn.close()
```

2) /api/countries/<code>/info

Mediante el método GET, la API se conecta a la base de datos y ejecuta la query de *"SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as currencyCode, moneda*

as currencyName, lengua as lang, nombre as name FROM info WHERE alpha2 = %s" puesto que la ruta requiere un código específico proveniente del código único alpha2 de un país, posterior a esto, guarda los datos y los transforma en formato json y los entrega.

```
@app.route('/api/countries/<code>/info', methods=['GET'])
def code(code):
        try:
                conn = mysql.connect()
                cursor = conn.cursor(pymysql.cursors.DictCursor)
                cursor.execute("SELECT alpha3 as abbr, alpha2 as code, codigo_
                country = cursor.fetchone()
                respone = jsonify(country)
                respone.status_code = 200
                return respone
        except Exception as e:
                print(e)
        finally:
                cursor.close()
                conn.close()
```

3) /api/indicators/<countryCode>/<indicatorCode>/<year>/info

Mediante el metodo GET, la API se conecta a la base de datos y ejecuta la query de *"SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as currencyCode, moneda as currencyName, lengua as lang, nombre as name FROM info WHERE alpha2 = countryCode"* para guardar los valores dentro de una variable llamada country y en base a que nuestra base de datos no tiene identificado el nombre del indicador como variable identificable, se crearon diferentes IF para cada uno de los indicadores correspondientes, dentro de este se realiza una nueva query "*SELECT indicador_name as value, anno as year FROM nameofindicator where anno =%s", year"* considerando el año que se está exigiendo, y esto se guarda en una nueva variable y se entrega un json con el nombre del indicador correspondiente, el código, la unidad de este y se referencia la variable country anteriormente definida, luego se actualiza el json con los valores de la última query y retorna el resultado, como muestra la imagen.

```
if (indicatorCode == 'PIB'):
    cursor.execute("SELECT indicador_pib as value, anno as year FROM pib where anno =%s", year)
    cosa = cursor.fetchone()
    L = {
        "code" : "PIB",
        "name" : "Producto Interno Bruto",
        "unit" : "$US",
        "country" : country,
    }
    L.update(cosa)
```

4) /api/indicators/info

Esta ruta funciona mediante el metodo POST, por lo que es necesario ingresar en el body los valores de “indicatorCode, endYear, countryCode, startYear” en formato json para que la API funcione correctamente, al ingresar los datos, esta realiza la query de *"SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as currencyCode, moneda as currencyName, lengua as lang, nombre as name FROM info WHERE alpha2 = countryCode"* con el fin de buscar los datos del pais que se solicita y guardarlos en una variable llamada country, luego se realiza un WHILE con el fin de que recorra los años que se encuentran entre el startYear y endYear que solicita el post y así entregue los valores solicitados, dentro de este while la api realiza la misma acción anterior de los IF con el fin de entregar los valores correspondientes a esos indicadores, una vez definido los IF, se suma como contador el valor de startYear y se agregan los datos dentro de un lista para posteriormente ser convertida a json y entregarla.

```
@app.route('/api/indicators/info', methods=['POST'])
def indicators_info():
    try:
        indicatorCode = request.json['indicatorCode']
        endYear = request.json['endYear']
        countryCode = request.json['countryCode']
        startYear = request.json['startYear']
        conn = mysql.connect()
        cursor = conn.cursor(pymysql.cursors.DictCursor)
        cursor.execute("SELECT alpha3 as abbr, alpha2 as code, codigo_pais_moneda as currencyCode, moneda as currency
        country = cursor.fetchone()
        annos = []
        while (startYear != endYear+1 and startYear <= endYear):
            if (indicatorCode == 'PIB'):
                cursor.execute("SELECT indicador_pib as value, anno as year FROM pib where anno =%s", startYear)
                cosa = cursor.fetchone()
                L = {
                    'code' : "PIB",
                    'name' : "Producto Interno Bruto",
                    'unit' : "$US",
                    'country' : country
                }
                L.update(cosa)
```

Finalmente y no menos importante, se define una condición donde el nombre de la instancia flask que dimos debe ser igual al main para que la api en python sea ejecutable y se definen el host y el puesto a utilizar.

```
if __name__ == "__main__":
    app.run(host='127.0.0.1', port=8080, debug=True)
```

Tecnología usada

El Lenguaje utilizado para el programa es Python en su versión 3.8

Python es una tecnología que tiene la capacidad de admitir métodos de programación estructurados y funcionales con un formato simple, una sintaxis muy definida y pocas keywords. En cuanto a sus librerías, estas son bastante grandes, variadas, portables y compatibles con Windows, Linux y Mac. Y lo mejor es que se trata de un lenguaje backend de código abierto (opensource), requisito que requiere el desarrollo de este proyecto.

Se utilizaron las librerías, Flask, Flask_HTTPAuth, PyMySQL, Datetime, Werkzeug.exceptions y security y Flask_Mysql

Flask es un microframework escrito en Python y concebido para facilitar el desarrollo de aplicaciones web, es decir, páginas web dinámicas, APIs, etc. bajo el patrón MVC.
Al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional, pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones (plugins) que se pueden instalar con Flask que le van dotando de funcionalidad.

Se utilizó la base de datos MySQL

MySQL es un sistema de gestión de base de datos de modelo cliente-servidor, utilizado mayormente para crear y administrar bases de datos relacionales. MySQL tiene como atrayente que es un servicio fácil de utilizar, bastante flexible, puede almacenar grandes cantidades de datos y es bastante seguro gracias a la necesidad de ingreso con usuario y contraseña.

Conclusión

Como conclusión, el trabajo llevo diferentes problemáticas ante los programadores causado por la poca experiencia referente al tema, no obstante, gracias a días de estudios y documentación, se llegó al resultado actual. Como programadores se consideró que el conocimiento básico de base de datos es sumamente importante para la creación de este tipo de API, puesto que no solo la conexión a la base de datos es necesaria, si no que el uso de los llamados de lectura de las base de datos, llamadas querys, son esenciales para obtener los datos de manera rápida y efectiva, el resto se basa en conocimientos básicos del uso de python como sus funciones While e If.

Actualmente, esta API solo se puede utilizar en la plataforma offline, es decir, no es escalable por el momento, quizás en algún futuro se podría implementar el servicio rest en alguna página web y sea utilizada con una mayor cantidad de datos con el fin de tener un mejor registro y solucionar mejor los valores.

No obstante, este trabajo nos adentro a mayor conocimiento en APIs para servicios de internet, como base a futuras actividades como ingenieros en computación e informática, REST se convirtió en un servicio simple y bastante lógico para estas actividades, además de poder ser trabajable con códigos abierto (Open Source) los cuales son bastante útiles para el aprendizaje de cualquier usuario que desea aprender y adentrarse en el mundo de la programación.