Laboratorio 16: Carrito de Compras con Redis y PostgreSQL

Prof. Heider Sanchez
ACLs: Ana María Accilio, Sebastián Loza

En este laboratorio desarrollarás un sistema web transaccional de carrito de compras, priorizando el rendimiento y la escalabilidad mediante el uso eficiente de caché. Utilizarás Flask como framework web, Redis como sistema de caché y PostgreSQL como base de datos persistente.

Objetivos

- Implementar un sistema de caché distribuido con Redis configurado en modo replicación (maestro-esclavo) para alta disponibilidad y tolerancia a fallos.
- Integrar Redis y PostgreSQL aplicando el patrón Cache-Aside, optimizando la gestión y consulta de datos para mejorar los tiempos de respuesta y reducir la carga sobre la base de datos.

Estructura del Proyecto

```
ecommerce/
├── app/
│   ├── __init__.py          # Inicialización de Flask
│   ├── config.py            # Configuraciones
│   ├── cache/               # Módulos de caché
│   ├── models/              # Modelos de datos
│   ├── routes/              # Rutas de la API
│   └── services/            # Servicios de negocio
├── scripts/
│   └── seed_data.py         # Script para datos de prueba
├── requirements.txt         # Dependencias
└── run.py                  # Punto de entrada
```

Pasos de Instalación

0. Verificar Versión de Python

Este proyecto requiere Python 3.12 o superior. Verifica tu versión:

```
python --version
```

1. Crear y Activar Entorno Virtual

Usando `venv`:

```
python -m venv venv
venv\Scripts\activate
```

Usando `conda`:

1. Crear un nuevo entorno conda:

    ```
    conda create -n ecommerce python=3.12
    ```

2. Activar el entorno:

    ```
    conda activate ecommerce
    ```

2. Instalar Dependencias

Usando `pip`:

```
pip install Flask==3.0.0
pip install redis==5.0.1
pip install psycopg2-binary==2.9.9
pip install Flask-SQLAlchemy==3.1.1
pip install python-dotenv==1.0.0
```

O usando el archivo `requirements.txt`:

```
pip install -r requirements.txt
```

Usando `conda`:

Instalar las dependencias principales con conda (las dependencias secundarias se instalan automáticamente):

```
conda install flask
conda install -c conda-forge redis-py
conda install psycopg2-binary
conda install flask-sqlalchemy
conda install -c conda-forge python-dotenv
```

5. Inicializar Base de Datos

1. Abrir pgAdmin o psql y ejecutar:

```
CREATE DATABASE ecommerce;
```

2. Insertar datos de prueba

```
python -m scripts.seed_data
```

6. Ejecutar la Aplicación

1. Activar entorno virtual (si no está activo):
   - Si usaste `venv`:
     ```
     venv\Scripts\activate
     ```
   - Si usaste `conda`:
     ```
     conda activate ecommerce
     ```
2. Ejecutar la aplicación:
   ```
   python run.py
   ```

> La API estará disponible en `http://localhost:5000`

7. Endpoints de la API

Método	Endpoint	Descripción
GET	`/cart/<user_id>`	Obtener carrito
POST	`/cart/<user_id>/add`	Agregar item
POST	`/cart/<user_id>/remove/<product_id>`	Eliminar item
PUT	`/cart/<user_id>/update/<product_id>`	Actualizar cantidad
POST	`/cart/<user_id>/clear`	Limpiar carrito

Ejemplos de Uso (Postman)

Obtener Carrito

- Método: GET
- URL: `http://localhost:5000/cart/user123`
- Headers: No requeridos

Agregar Item

- Método: POST
- URL: `http://localhost:5000/cart/user123/add`
- Headers:
  - Content-Type: application/json
- Body (raw JSON):

```
{
    "product_id": 1,
    "name": "Laptop",
    "price": 999.99,
    "quantity": 1
}
```

Actualizar Cantidad

- Método: PUT
- URL: `http://localhost:5000/cart/user123/update/1`
- Headers:
  - Content-Type: application/json
- Body (raw JSON):

```
{
    "quantity": 2
}
```

Instalación de Redis

Para instalar y ejecutar Redis, usaremos Docker:

1. Iniciar el servidor Redis (standalone):

```
docker run --name redis-ecommerce -p 6379:6379 -d redis
```

2. Comandos útiles para administrar el servidor:

```
# Detener el servidor Redis
docker stop redis-ecommerce

# Reiniciar el servidor Redis
docker start redis-ecommerce
```

Parte Evaluada

Requerimientos

1. **(3 pts) Configuración de Redis en Modo Replicación**
   - Utilizar Docker Compose para desplegar Redis en modo replicación: 1 nodo maestro y 2 nodos esclavos
   - Documentar la configuración de replicación
   - Evidenciar la distribución de datos entre nodos
2. **(3 pts) Datos de Prueba**
   - Modificar seed_data.py para manejar mas datos de prueba.
   - Al menos 20 registros en tabla DBCart, y cada carrito debe contener entre 3 y 10 items (DBCartItem)

- Los items deben tener información variada y realista:
  - Nombres de productos descriptivos
  - Precios variados (entre $1 y $1000)
  - Cantidades diferentes (entre 1 y 5 unidades por item)

3. **(10 pts) Implementación del Patrón Cache-Aside**
   - Implementar el patrón para las operaciones CRUD del carrito utilizando Redis como caché.
   - **Tiene que identificar cuando se debe consultar Redis y cuando PostgreSQL**.
   - Establecer un tiempo de expiración de 30 minutos para los carritos en caché
   - Implementar un endpoint `/stats/top-products` que utilice Redis como caché para retornar el top 10 de productos más comprados.
   - Garantizar la consistencia entre Redis y PostgreSQL
4. **(4 pts) Pruebas de Rendimiento**
   - Documentar tiempos de respuesta con y sin caché (produzca un entorno favorable).
   - Evidenciar la distribución de carga entre nodos Redis

Entregables

1. Código fuente con la implementación completa
2. Documentación de la configuración de Redis en modo replicación
3. Evidencias de las pruebas de rendimiento
4. Informe técnico detallando la implementación del patrón Cache-Aside en el sistema.