Progetto WEB - Migrazione dati

Introduzione

Il progetto prevede la migrazione di dati da un database remoto a uno locale utilizzando un sistema composto da tre servizi: un webservice remoto in PHP, un intermediario in Java e un webservice locale in Python/Django. L'obiettivo è garantire un trasferimento sicuro e generalizzato dei dati in formato JSON.

Analisi

Principi di funzionamento

- **Esportazione della struttura del DB**: Il database locale non necessita che venga precedentemente definito lo schema delle tabelle, il servizio è in grado di esportare anche il create statement delle tabelle così da non incorrere in problemi quali la possibile disomogeneità tra schemi delle due tabelle anche in presenza di modificazioni allo schema delle tabelle del database remoto.
- **Formato comune**: L'esportazione e l'importazione delle tabelle avvengono tramite un formato comune tra il servizio PHP e quello PYTHON, nel nostro caso è stato scelto un formato JSON strutturato ad hoc. Questo consente una migrazione più lineare e una minore elaborazione dei dati in quanto non ne è richiesta la traduzione da un formato ad un altro.
- **Sicurezza**: Il server PHP dichiara esplicitamente le tabelle esportabili in modo che si eviti l'esposizione dell'intero database, così facendo si previene l'esportazione di dati che si presuppone non siano accessibili a chiunque.
- **Protocollo e formato**: Tutti i servizi operano con protocollo HTTP e le risposte sono in formato JSON. Questo rende la comunicazione più semplice e diretta tra i diversi web services.

Architettura

Il sistema è composto da tre componenti principali:

1. **PHP Web Service**: Esporta i dati e la struttura delle tabelle.
2. **Java Web Service**: Funziona da intermediario, comunicando con il webservice PHP e inoltrando i dati al servizio Python.
3. **Python Web Service**: Importa i dati e crea le tabelle nel database locale.

Di seguito verranno descritti nel dettaglio.

- **PHP Web Service**
  - **Struttura progetto**:
    - `api/` : cartella contenente gli endpoint esposti
    - `lib/` : cartella contenente le librerie usate
    - `config.php` : file di configurazione del servizio in cui vengono dichiarate le tabelle che il servizio può esportare.
  - **Funzionamento**:
    - `GET` - **`/api/getTables.php`** : restituisce le tabelle esportabili in formato json
    - `GET` - ** `/api/getTableData.php?table={table}` **: riceve un parametro table, se esiste la tabella richiesta, genera il create statement associato, legge le righe e le colonne e restituisce i dati in formato JSON (vedi `php-service/response.json`). Utilizziamo un'unica API per richiedere tutte le tabelle tramite un parametro che contiene il nome della tabella desiderata presente all'interno del DB (il nome della tabella viene ricavato dalla precedente richiesta effettuata da `/api/getTables.php`). Mediante apposita funzione lo statement SQL per la creazione della tabella viene convertito in sintassi postgres.

- **JAVA Web Service**
  - **Struttura progetto**:

    Progetto Tomcat standard
  - **Funzionamento**:
    - `GET` - **`/MigrateTable?table={table}`** : riceve un parametro table, contatta il servizio php /api/getTableData.php specificando la tabella richiesta e salvo errori, invia il risultato al servizio python.
    - `GET` - ** `/FullMigration` **: legge la lista di tabelle esportabili dal servizio php mediante l'api "/api/getTables.php" e per ogni tabella esegue lo stesso comportamento dell'endpoint `/MigrateTable`

- **PYTHON Web Service**
  - **Struttura progetto**:

    Progetto Django standard
  - **Funzionamento**:

        - `POST` - **`/importTable`**: riceve un JSON nel formato ad hoc, controlla se la tabella da importare esiste già, se non esiste viene creata eseguendo il create statement inviato, successivamente importa i dati nella tabella

Istruzioni

- **php-service**

  Contiene il web-service caricato sul server altervista all'url: `https://uvette.altervista.org/php-service/api`

- **java-service**

  Contiene il web-service tomcat

  - **Installazione**: copiare l'intera cartella `java-service` nella cartella `webapps` del server Tomcat, se necessario ricompilare i file `.java` nella cartella `WEB-INF/classes/` con il comando

    ```
    javac  -cp "./WEB-INF/lib/*" ./WEB-INF/classes/*
    ```

  - **Configurazione**: modificare se necessario i due url dei servizi nel file `WEB-INF/web.xml` `(riga 16 - 20)`

- **python-service**

  Contiene il web-service django

  - **Installazione**: installare se necessario i moduli nel file `requirements.txt`, avviare il server django con il comando:

    ```
    python .\manage.py runserver
    ```

  - **Configurazione**: modificare i parametri di connessione al database postgres nel file `python_service/settings.py` `(riga 74)`