Implementazione di uno strumento per la migrazione di dati

La directory di lavoro (*code*) vede la presenza dei seguenti tre progetti:

- *source-service*, deputato all'implementazione del web service remoto;
- *middleware-service*, che ospita il layer intermedio dell'applicazione;
- *destination-service*, il quale si occupa di gestire il web service locale.

Ad orchestrare le operazioni è il secondo, che ricorre a Maven non solo per importare le dipendenze di cui si serve, ma anche al fine di creare un file WAR in grado di "confezionare" l'intera web app Java, in modo tale che la si possa trasferire/condividere agevolmente.

È stata introdotta una JSP (in sostanza, una pagina HTML) a supporto della servlet, così che l'utente abbia a disposizione un'interfaccia grafica intuitiva per comunicare alla stessa quali tabelle del database remoto desideri migrare verso quello locale; ciò è possibile grazie ad uno script JS a cui la pagina si appoggia, il quale effettua richieste, oltre che alla servlet, ad uno script PHP lato server per recuperare la lista di tabelle afferenti al database MySQL.

All'interno di *destination-service* si possono trovare, invece, un file da eseguire al fine di lanciare in locale un server Python ed una cartella denominata *app*; all'interno di quest'ultima è di particolare interesse il file *config.py*, il quale ospita un dizionario che definisce i parametri di connessione alla base di dati Postgres.

Dal punto di vista tecnico, il web service Python ricorre a Flask per configurare un *endpoint* atto a ricevere richieste dalla servlet e restituire risposte in formato JSON (il quale viene utilizzato nel contesto dell'intero applicativo), oltre che a Psycopg per connettersi al database locale ed effettuare operazioni su quest'ultimo.

Lo strato intermedio, d'altra parte, utilizza Unirest per indirizzare richieste HTTP ai web service e gestirne le risposte, oltre a Gson per poter strutturare correttamente eventuali messaggi di errore destinati alla JSP da cui è affiancato; servlet e UI comunicano, infatti, per mezzo di un oggetto **`PrintWriter`**, il quale viene istanziato a partire dall'oggetto che rappresenta la risposta fornita dalla servlet (**`HttpServletResponse`**).

Ricapitolando, gli step da seguire per testare la web app sono i seguenti:

1) Sostituire, all'interno del file *config.py* (nella cartella *app* di *destination-service*), la password di accesso a PostgreSQL con la propria (scelta in fase di installazione);

2) lanciare il server locale Python (che si "mette in ascolto" sul port 5000), eseguendo lo script *run.py* all'interno di *destination-service*;

3) inserire il file *middleware-service-1.0-SNAPSHOT.war* nella cartella *webapps* della *Tomcat 9.0 Program Directory*, la cui posizione dipende da quanto selezionato in fase di installazione: nel mio caso C:\Program Files\Apache Software Foundation\Tomcat 9.0;

4) accedere a questo indirizzo messo a disposizione per mezzo di Tomcat per iniziare ad interagire con l'applicativo.