Workflow di esecuzione

Compilato il template della richiesta in base allo scenario di utilizzo e alla query scelti seguire i seguenti passaggi

ChatGPT e Google Bard

1. Aprire una nuova chat nell'interfaccia dello strumento
2. Copiare la richiesta ottenuta a partire dal template ed inoltrare la richiesta allo strumento
   2.1. Prima di inoltrare la richiesta avviare un cronometro e, una volta terminata la generazione della risposta, annotare il tempo impiegato alla voce "***Tempo generazione risposta***" della tabella delle metriche. (!!! Il cronometro va stoppato quando il cursore di generazione della risposta si ferma !!!)
3. Controllare la risposta generata per assicurarsi che il codice sia completo.
   3.1. Qualora non fosse completo chiedere allo strumento di completare la generazione del codice ed incrementare di 1 il valore "***Numero di Modifiche con ausilio dello strumento***" della tabella delle metriche.
4. Copiare il codice generato in un nuovo Notebook Jupyter, aggiungendo in cima alla cella del codice il comando *%%timeit -r 10 -n 30* per calcolare il tempo medio di esecuzione del codice.
   4.1. Qualora servisse, modificare anche i parametri per la connessione col database.
5. Eseguire il codice.
   5.1. ***Se l'esecuzione ha prodotto errori***, incrementare di 1 il valore "***Errori***" della tabella delle metriche, controllare l'errore e:
      5.1.1. Se l'errore si può correggere facilmente, correggerlo ed incrementare di 1 il valore "***Numero di Modifiche senza ausilio dello strumento***" della tabella delle metriche.
      5.1.2. Altrimenti copiare l'errore e farlo presente allo strumento chiedendo di risolverlo. In questo caso incrementare di 1 il valore "***Numero di Modifiche con ausilio dello strumento***" della tabella delle metriche.
      5.1.3. Apportare le modifiche suggerite e ripetere il punto 5.
   5.2. ***Se l'esecuzione non ha prodotto errori***, annotare il tempo medio di esecuzione del codice nel capo "***Tempo di esecuzione dello script***" nella tabella delle metriche e salvare il file.
6. Salvare il Notebook Jupyter come "*TipologiaQuery_Dialetto_StrumentoUtilizzato_ScenarioUtilizzo.ipynb" ed* effettuare il commit del file nel repository github del progetto.

Github Copilot

1. Aprire un nuovo Jupyter Notebook in Visual studio code, aggiungendo all'inizio il comando %%timeit -r 10 -n 30
2. Copiare tra tripli apici (''' … ''') la richiesta ottenuta a partire dal template, posizionare il cursore all'esterno del blocco commento ed attendere la generazione del codice dopo aver avviato un cronometro.
   2.1. Accettare i suggerimenti dati dallo strumento con il tasto *Tab*
   2.2. Qualora lo strumento non dovesse generare più codice premere il tasto invio ed attendere nuovi suggerimenti
3. Alla fine della generazione del codice, stoppare il cronometro ed annotare il tempo impiegato alla voce "***Tempo generazione risposta***" della tabella delle metriche.
4. Controllare se il codice generato segue la struttura indicata e, qualora servisse, modificare i parametri per la connessione col database.
5. Eseguire il codice.
   5.1. ***Se l'esecuzione ha prodotto errori***, incrementare di 1 il valore "***Errori***" della tabella delle metriche, controllare l'errore e:

5.1.1. Se l'errore si può correggere facilmente, correggerlo ed incrementare di 1 il valore "***Numero di Modifiche senza ausilio dello strumento***" della tabella delle metriche.
5.1.2. Altrimenti copiare l'errore e, dopo aver posizionato il cursore sulla riga che lo genera, farlo presente allo strumento grazie alla combinazione di tasti *ctrl + i*. In questo caso incrementare di 1 il valore "***Numero di Modifiche con ausilio dello strumento***" della tabella delle metriche.
5.1.3. Apportare le modifiche suggerite e ripetere il punto 5.

5.2. ***Se l'esecuzione non ha prodotto errori***, annotare il tempo medio di esecuzione del codice nel capo "***Tempo di esecuzione dello script***" nella tabella delle metriche e salvare il file.

6. Salvare il Notebook Jupyter come "*TipologiaQuery_StrumentoUtilizzato_ScenarioUtilizzo.ipynb" ed* effettuare il commit del file nel repository github del progetto.

Template richieste

Sulla base dei tre scenari di utilizzo, sono stati definiti dei template da seguire. Questi modelli predefiniti rappresentano uno schema che fornisce una struttura organizzata per formulare richieste specifiche e dettagliate che aiutano a garantire che le richieste siano chiare, complete e pertinenti al contesto.

Tutti questi template seguono uno schema prefissato che è il seguente:

- **Nome della Richiesta**: Breve nome per identificare in modo univoco la richiesta
- **Descrizione del Problema**: Spiegazione chiara e concisa del problema su cui stai lavorando.
- **Query da Tradurre**: Specificare la query SQL che si desidera tradurre.
  - **Solo per le richieste fatte a Github Copilot**: includere parti di codice o pseudocodice correlato alla richiesta.
- **Dettagli Aggiuntivi**: Includere eventuali dettagli aggiuntivi che possono essere utili per comprendere meglio il contesto della richiesta, come ad esempio la tipologia di database utilizzato, l'uso o meno di funzioni, la gestione delle eccezioni, il livello di dettaglio dei commenti.
- **Requisiti Tecnici**: Fornire dettagli sui requisiti tecnici specifici, come l'utilizzo di determinate librerie o framework, la suddivisione della traduzione in moduli separati o l'uso di un determinato design pattern.
- **Livello di Sicurezza**: Specificare il livello di sicurezza richiesto per il codice generato. Ad esempio, se è necessario implementare misure di sicurezza come la prevenzione di SQL injection, gestione sicura delle password, ecc.
- **Extra**: Possibilità di specificare alcuni extra nella richiesta come, ad esempio, fornire una breve descrizione del comportamento atteso, un esempio di output generato oppure la possibilità di generare dei test per verificare il corretto funzionamento del codice.

A seconda dello scenario di utilizzo dello strumento è possibile compilare tutti o solo alcuni di questi campi.

Mostriamo ora un esempio di richiesta per ognuno degli strumenti utilizzati e per ognuno degli scenari di utilizzo, seguendo il template appena illustrato. Per semplicità viene selezionata la stessa query per ognuna delle richieste.

Esempi

Esempio richiesta a ChatGPT, Scenario di Utilizzo Base

Nome della Richiesta	Traduzione query Select
Descrizione del Problema	Devo tradurre la query di select dal linguaggio sql al python

Query da Tradurre	`SELECT *`
	`FROM ApparecchiatureElettroniche`
	`WHERE CAST (SUBSTRING_INDEX(RisoluzioneSchermo, 'x', -1) AS SIGNED) > 1080`
	`ORDER BY DataProduzione DESC;`

Dettagli Aggiuntivi	Il database utilizzato è un database MySQL

Requisiti Tecnici	-
Livello di Sicurezza	-
Extra	-

Esempio richiesta a Google Bard, Scenario di Utilizzo Avanzato

Nome della Richiesta	Traduzione query Select
Descrizione del Problema	Devo tradurre la query di select dal linguaggio sql al python

Query da Tradurre	`SELECT *`
	`FROM ApparecchiatureElettroniche`
	`WHERE CAST (SUBSTRING_INDEX(RisoluzioneSchermo, 'x', -1)`
	`AS SIGNED) > 1080`
	`ORDER BY DataProduzione DESC;`

Dettagli Aggiuntivi	Il database utilizzato è un database MySQL, voglio suddividere il codice in funzioni e gestisci in maniera appropriate le eccezioni. Inoltre, commenta adeguatamente ogni funzione del codice.
Requisiti Tecnici	Utilizza la libreria mysql-connector, adotta un pattern di tipo Model-View-Controller (MVC) e suddividi tutto il codice in moduli separati tra di loro.
Livello di Sicurezza	Voglio che nel codice vengano adottate misure di sicurezza per prevenire attacchi di SQL injection. Inoltre, voglio che dati sensibili non vengano inseriti direttamente nel codice.
Extra	Voglio che l'output del codice venga mostrato con una struttura tabellare. Organizza i vari moduli in directory aggiungendo anche quella per i test. Genera alcuni test.

Esempio richiesta a Github Copilot, Scenario di Utilizzo Intermedio

Nome della Richiesta	Traduzione query Select
Descrizione del Problema	Devo tradurre la query di select dal linguaggio sql al python

Query da Tradurre	`SELECT *`
	`FROM ApparecchiatureElettroniche`
	`WHERE CAST (SUBSTRING_INDEX(RisoluzioneSchermo, 'x', -1)`
	`AS SIGNED) > 1080`
	`ORDER BY DataProduzione DESC;`

Dettagli Aggiuntivi	Il database utilizzato è un database MySQL, voglio suddividere il codice in funzioni e commenta il codice.
Requisiti Tecnici	Utilizza la libreria mysql-connector.
Livello di Sicurezza	Fai attenzione alla sicurezza dei dati.
Extra	-
Struttura del codice	Struttura del codice da seguire #import delle librerie #definizione di variabili #funzione per connettersi al database #funzione per eseguire la query #funzione per stampare i risultati #main

Scenari Di Utilizzo

Scenario Base

Descrizione: In questo scenario, la richiesta fatta allo strumento e di quelle più basilari possibili. Vengono condivise poche informazioni con lo strumento, come ad esempio il database utilizzato. Lo scopo è quello di ottenere un codice almeno funzionante.

Risposta attesa: Il codice generato dallo strumento è molto basilare, è formato da un unico script al cui interno viene gestita la logica di connessione al database, l'esecuzione della query e la stampa a video dei risultati.

Scenario Intermedio

Descrizione: In questo scenario, la richiesta fatta allo strumento è più articolata. Le informazioni che si condividono con lo strumento sono più precise e dettagliate e lo scopo è quello di ottenere non soltanto un codice funzionante, ma anche ben strutturato.

Risposta attesa: Il codice generato dallo strumento è suddiviso in funzioni ognuna delle quali si occupa di un'operazione ben definita, come ad esempio la connessione al database o l'esecuzione della query. Alla fine, le varie funzioni verranno richiamate in un blocco main che ne permetterà l'esecuzione complessiva. Ci si aspetta anche che i commenti generati spieghino in maniera esaustiva le varie parti del codice generato.

Scenario Avanzato

Descrizione: In questo scenario, la richiesta fatta allo strumento è ancora più articolata rispetto allo scenario precedente. Le informazioni condivise con lo strumento non serviranno solamente a generare un codice ben strutturato, ma anche il più sicuro, leggibile e manutenibile possibile. In questo scenario è possibile anche ipotizzare che l'utilizzatore dello strumento possa chiedere di voler generare un'interfaccia utente che permetta di eseguire la query o di voler testare il codice.

Risposta attesa: Il codice generato dallo segue una struttura a layer, ognuno dei quali contiene la logica di programmazione di una parte specifica dell'applicazione. Vi è una chiara separazione tra la logica di dominio, la logica di applicazione, la configurazione del database e, in maniera opzionale, l'interfaccia utente e i test. Questo approccio favorisce la manutenibilità, la leggibilità e la scalabilità del codice, consentendo di apportare modifiche o estensioni in modo più agevole. Inoltre, la separazione delle informazioni di connessione al database contribuisce a migliorare la sicurezza del codice, evitando l'inclusione diretta di password nel sorgente.