Descrizione del Progetto: Gestione CSV con FastAPI e Docker

Parte 1: Configurazione di Anaconda, GitHub e Visual Studio Code

1. **Configurazione di Anaconda:**
    - Installare Anaconda sul tuo sistema.
    - Creare un environment chiamato 'project' utilizzando Anaconda per gestire le dipendenze del progetto.
2. **Creazione di un Progetto su GitHub:**
    - Creare un nuovo repository su GitHub chiamato 'csv-crud-fastapi-docker'.
    - Configurare il repository con un README iniziale.
3. **Configurazione del Progetto su Visual Studio Code:**
    - Clonare il repository GitHub sul tuo computer utilizzando Visual Studio Code.
    - Configurare l'environment 'project' di Anaconda come ambiente Python per il progetto in Visual Studio Code.
    - Creare un file `requirements.txt` contenente le dipendenze del progetto Python.

Parte 2: Implementazione delle operazioni CRUD con FastAPI su un CSV

1. **Implementazione in Python con FastAPI:**
    - Utilizzare Python e FastAPI per implementare le operazioni CRUD (Create, Read, Update, Delete) su un file CSV.
    - Il CSV deve contenere le seguenti colonne: ID, nome, cognome, Codice fiscale.
    - Creare un endpoint per ognuna delle operazioni CRUD utilizzando FastAPI.
        - Endpoint per la creazione di un nuovo record:
            - Metodo: POST
            - Path: `/items/`
        - Endpoint per ottenere tutti i record:
            - Metodo: GET
            - Path: `/items/`
        - Endpoint per ottenere un singolo record basato sull'ID:
            - Metodo: GET
            - Path: `/items/{id}`
        - Endpoint per aggiornare un record esistente:
            - Metodo: PUT
            - Path: `/items/{id}`
        - Endpoint per eliminare un record esistente:
            - Metodo: DELETE
            - Path: `/items/{id}`
        - Endpoint per ottenere il numero di righe nel CSV:
            - Metodo: GET
            - Path: `/items/count`

Endpoint per creare un nuovo record

Richiesta (POST):

```
curl -X POST "http://localhost:8000/items/" -H "Content-Type:
application/json" -d '{
  "id": 1,
  "nome": "Mario",
  "cognome": "Rossi",
  "codice_fiscale": "RSSMRA01A01H501A"
}'
```

Risposta attesa:

json

```
{
  "id": 1,
  "nome": "Mario",
  "cognome": "Rossi",
  "codice_fiscale": "RSSMRA01A01H501A"
}
```

Endpoint per ottenere tutti i record

Richiesta (GET):

```
curl -X GET "http://localhost:8000/items/"
```

Risposta attesa:

json

```
[
  {
    "id": 1,
    "nome": "Mario",
    "cognome": "Rossi",
    "codice_fiscale": "RSSMRA01A01H501A"
  }
]
```

Endpoint per ottenere un singolo record basato sull'ID

Assumendo che l'ID del record creato sia 1:

Richiesta (GET):

```
curl -X GET "http://localhost:8000/items/1"
```

Risposta attesa:

json

```
{
  "id": 1,
  "nome": "Mario",
  "cognome": "Rossi",
  "codice_fiscale": "RSSMRA01A01H501A"
}
```

Endpoint per aggiornare un record esistente

Assumendo che si voglia aggiornare il record con ID 1:

Richiesta (PUT):

```
curl -X PUT "http://localhost:8000/items/1" -H "Content-Type:
application/json" -d '{
  "id": 1,
  "nome": "Luigi",
  "cognome": "Verdi",
  "codice_fiscale": "VRDLGI01A01H501A"
}'
```

Risposta attesa:

json

```
{
  "id": 1,
  "nome": "Luigi",
  "cognome": "Verdi",
  "codice_fiscale": "VRDLGI01A01H501A"
}
```

Endpoint per eliminare un record esistente

Assumendo che si voglia eliminare il record con ID 1:

Richiesta (DELETE):

```
curl -X DELETE "http://localhost:8000/items/1"
```

Risposta attesa:

json

```
{
  "message": "Item deleted successfully"
```

```
}
```

Endpoint per ottenere il numero di righe nel CSV

Richiesta (GET):

```
curl -X GET "http://localhost:8000/items/count"
```

Risposta attesa:

```
{

  "count": 0  // Questo valore dipenderà dal numero di record presenti nel CSV

}
```

Parte 3: Creazione di un'Immagine Docker per il Web Service

1. **Creazione di un'Immagine Docker:**
    - Creare un Dockerfile per il progetto FastAPI.
    - Utilizzare il Dockerfile per creare un'immagine Docker che includa il web service realizzato con FastAPI e tutte le sue dipendenze.

Parte 4: Testing degli Endpoint in Docker con Postman

1. **Testing degli Endpoint con Postman:**
    - Utilizzare Postman per testare tutti gli endpoint del web service eseguito tramite l'immagine Docker.
    - Verificare che le operazioni CRUD e l'endpoint per il conteggio delle righe funzionino correttamente.