DOCUMENTO TECNICO DI PROGETTO

Titolo progetto: Analisi del settore della pesca in Italia

Candidato: [Cognome Nome]

Data: [Data dell'esame]

Corso: Fintech Software Developer – ITS ICT Piemonte

1. Obiettivo del progetto

L'obiettivo del progetto è la realizzazione di un'applicazione software per:
- importare, pulire e normalizzare dati pubblici sul settore della pesca in Italia;
- calcolare alcune metriche statistiche annuali;
- rendere disponibili i dati e le metriche tramite un'interfaccia web API;
- (facoltativo) visualizzare i dati in forma grafica.

2. Tecnologie utilizzate

- Linguaggio di programmazione: Python 3
  Semplice, multipiattaforma, adatto per data analysis e web API.
- Librerie principali:
  • pandas: per la gestione dei dati tabellari.
  • flask: per la creazione delle API web.
  • matplotlib: per la visualizzazione dei dati (grafici).
- Database: SQLite
  Scelto per semplicità, leggerezza e facilità d'integrazione con Python.

3. Struttura del progetto

- csv_import.py: importazione e salvataggio locale dei dati CSV.
- create_db.py: creazione e popolamento del database SQLite.
- series.py: calcolo delle metriche statistiche richieste dalla traccia.
- app.py: applicazione Flask per l'esposizione delle API.
- graph.ipynb: notebook per la visualizzazione grafica (facoltativo).
- csv/: directory contenente i file CSV scaricati.
- pesca.db: database SQLite.
- documentazione_progetto_pesca.pdf: documentazione scritta.
- createVenv: script per creare un ambiente virtuale.
- .gitignore, Estensioni VSCODE.txt: file di supporto.

4. Analisi del problema e scelte progettuali

a. Importazione dati
I dati sono stati scaricati in formato .csv da fonti open data (datiopen.it) utilizzando

pandas.read_csv(), salvandoli localmente nella cartella csv/. Si è scelto di separare l'importazione in uno script autonomo (csv_import.py) per modularità e riusabilità.

b. Normalizzazione
I dataset contenevano anni mancanti. Si è deciso di utilizzare interpolazione lineare tramite pandas.DataFrame.interpolate(). Motivazione: semplice da implementare e sufficiente per ricostruire valori mancanti su base temporale.

c. Database
È stato scelto SQLite per semplicità d'uso, essendo integrato in Python. Motivazione: per un'applicazione monopostazione o da esame è più che sufficiente e non richiede installazione di server esterni.

d. Calcolo delle metriche
Il file series.py calcola le metriche richieste:
1. Produttività totale per area geografica
2. Produttività totale nazionale
3. Media percentuale del valore aggiunto pesca/piscicoltura per area
4. Media variazione percentuale occupazione nazionale
5. Media variazione percentuale occupazione per area

Per l'aggregazione sono state usate le macro-aree indicate nella traccia, associate a un dizionario Python. I dati sono raggruppati per anno con groupby() e salvati nel database. Scelta non ovvia: è stata effettuata una media semplice delle percentuali, come richiesto nella nota metodologica della traccia, senza ponderare per popolazione.

e. API REST
Nel file app.py è stata sviluppata un'applicazione Flask che espone i dati e le serie calcolate tramite API HTTP. Sono stati previsti parametri di filtro per anno (?da=XXXX&a=YYYY) come richiesto.
Esempio endpoint: /api/produttivita_per_area?da=2010&a=2020

f. Grafici
Nel file graph.ipynb è stato creato un esempio di visualizzazione con matplotlib, utile per analisi esplorative o dashboard future.

5. Considerazioni aggiuntive

- Assunzioni fatte:
  - Si assume che i dati siano affidabili, anche se presentano valori mancanti.
  - L'interpolazione è sufficiente per integrare i dati temporali.
  - I nomi delle regioni nei CSV coincidono con quelli del dizionario delle macro-aree.
- Scelte di semplicità:

- Uso di SQLite per evitare complicazioni con installazioni server.
- API essenziali e documentate solo per i dataset richiesti.

6. Estensioni possibili (non richieste)

- Interfaccia grafica web con Flask + HTML/CSS
- Frontend React o Vue.js con chiamate API
- Database PostgreSQL o MySQL per produzione
- Dashboard interattiva con Streamlit o Dash

7. Conclusione

L'applicazione sviluppata risponde ai requisiti richiesti dalla traccia in modo modulare e semplice, con strumenti open source, mantenendo leggibilità e chiarezza nel codice. È facilmente estendibile e permette di replicare l'analisi su dataset simili, anche al di fuori del dominio "pesca".

8. Allegati

- Progetto completo: Prova2Python-main/
- Database SQLite: pesca.db
- Documentazione Jupyter: graph.ipynb
- Codice API: app.py
- Documento tecnico: documentazione_progetto_pesca.pdf