Esercitazione: Creazione di un sistema di gestione di un'autofficina con architettura 3-tier

Obiettivo: Creare un sistema di gestione per un'autofficina utilizzando un'architettura 3-tier, composto da:

- **Tier 1 (Database)**: SQL Server per la gestione dei dati.
- **Tier 2 (Backend API)**: Un'API in ASP.NET che gestisce la logica e la comunicazione con il database.
- **Tier 3 (Frontend)**: Un'interfaccia utente realizzata con HTML, CSS e JavaScript per interagire con l'API e visualizzare i dati.

Requisiti del sistema:

Il sistema dovrà gestire **veicoli** e **clienti**. I **veicoli** avranno le seguenti caratteristiche:

- **Codice Veicolo**: Generato automaticamente dal sistema.
- **Targa**.
- **Modello**.
- **Marca**.
- **Anno di immatricolazione**.
- **Prezzo dell'intervento**.
- **Stato dell'intervento**: (es. in corso, completato, da fare).
- **Data di ingresso in officina**.

La seconda tabella dovrà gestire i **clienti**, con le seguenti caratteristiche:

- **Codice Cliente**: Generato automaticamente dal sistema.
- **Nome**.
- **Cognome**.
- **Indirizzo**.
- **Numero di telefono**.
- **Email**.

Ogni cliente può avere uno o più veicoli, quindi ci sarà una relazione tra le due tabelle (Cliente-Veicolo).

Funzionalità richieste:

1. **Gestione CRUD (Create, Read, Update, Delete) per i veicoli e i clienti**.
2. **Collegamento Cliente-Veicolo**: Ogni cliente può essere collegato a uno o più veicoli. Bisogna poter visualizzare i veicoli associati a ciascun cliente e viceversa.
3. **Aggiornamento dello stato dell'intervento**: Consentire l'aggiornamento dello stato dell'intervento per ciascun veicolo (ad esempio, passare da "in corso" a "completato").

4. **Modifica del prezzo dell'intervento con pulsanti**: Nella tabella dei veicoli, accanto a ogni riga, dovranno essere presenti dei pulsanti "+" e "-" per incrementare o decrementare il prezzo dell'intervento.
5. **Filtraggio dinamico dei veicoli**: Implementare un campo di input che consenta di filtrare dinamicamente i risultati della tabella dei veicoli in base ai criteri inseriti (senza utilizzare un intervallo di tempo per l'aggiornamento dei risultati).
6. **Totale degli interventi per stato**: All'inizio della tabella dei veicoli, visualizzare il totale degli interventi per ciascun stato (in corso, completato, ecc.).

Implementazione:

- **SQL Server (Database Layer)**: Creare due tabelle, una per i veicoli e una per i clienti, con una relazione one-to-many per collegare i clienti ai loro veicoli.
- **ASP.NET API (Business Logic Layer)**: Implementare un'API che consenta di effettuare le operazioni CRUD sui veicoli e sui clienti, e gestire le richieste di aggiornamento dello stato degli interventi.
- **Frontend HTML/JavaScript (Presentation Layer)**: Creare un'interfaccia utente che consumi le API e permetta di interagire con i dati, visualizzando e filtrando veicoli e clienti.