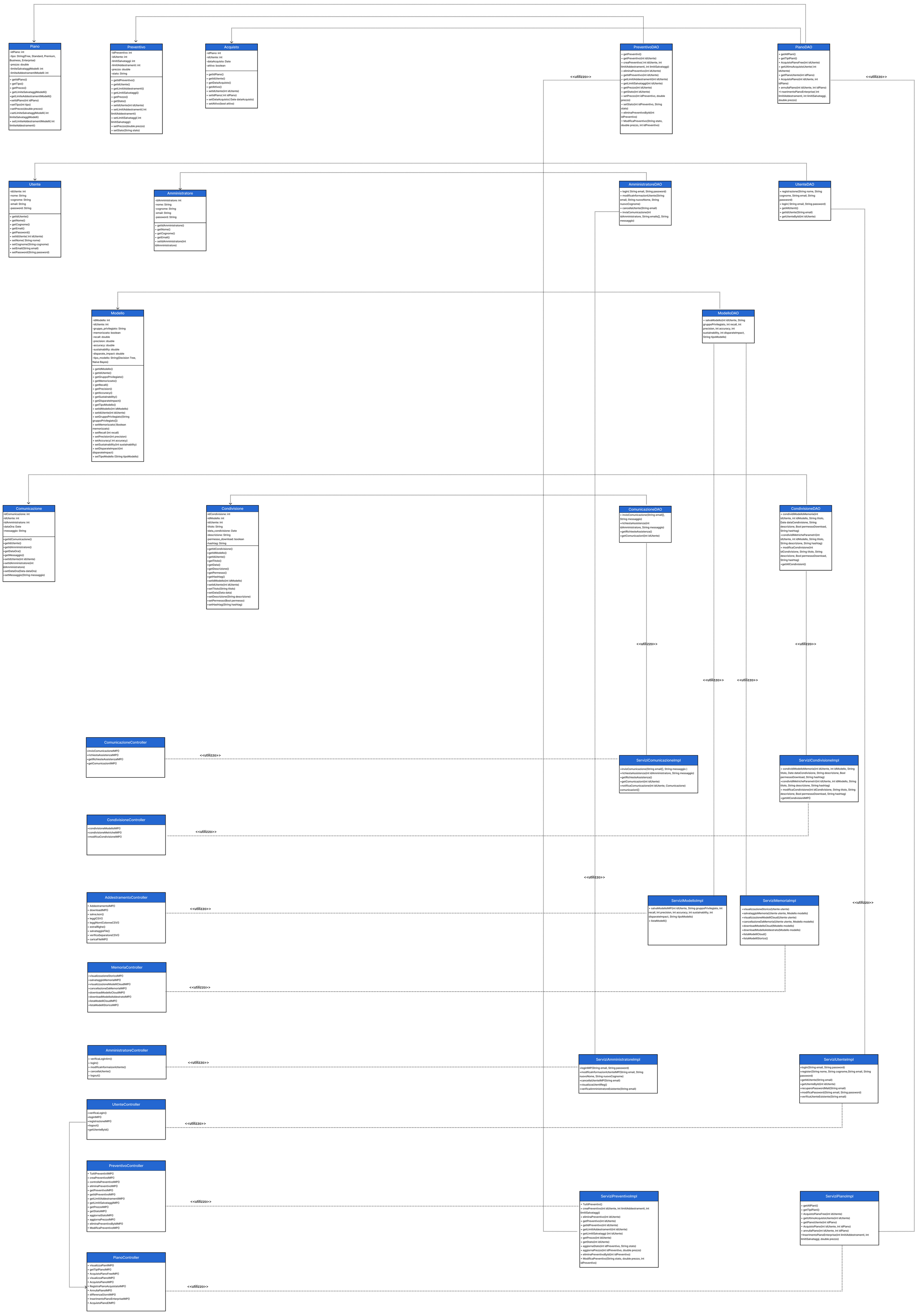

le relazioni sono di dipendenza e di utilizzo

quelle di dipendenza:
UtenteDAO contiene metodi che manipolano oggetti di tipo Utente o interagiscono con il database utilizzando oggetti di tipo Utente. Tuttavia, l'UtenteDAO potrebbe esistere e svolgere le sue funzioni anche senza dover istanziare direttamente oggetti Utente.
Ecco un razionale più dettagliato:
1. Utilizzo dei metodi del DAO: La classe UtenteDAO contiene metodi che effettuano operazioni sul database in relazione agli oggetti Utente. Ad esempio, il metodo login cerca un utente nel database e restituisce un oggetto Utente. La presenza di tali metodi implica una dipendenza dalla classe Utente.
2. Manipolazione di oggetti Utente: Anche se UtenteDAO non istanzia direttamente oggetti Utente, i suoi metodi lavorano con dati di tipo Utente, come nell'esempio del metodo login che crea un oggetto Utente basato sui dati ottenuti dal database.
3. Possibile evoluzione del sistema: Nel caso in cui il sistema evolva e nuovi requisiti richiedano manipolazioni più complesse sugli oggetti Utente, la dipendenza consentirebbe una maggiore flessibilità nell'estendere la logica senza dover modificare direttamente la classe UtenteDAO.
4. Separazione delle responsabilità: La dipendenza consente una separazione delle responsabilità, dove la classe UtenteDAO si occupa delle operazioni di accesso al database e la classe Utente si occupa della rappresentazione degli utenti. Questo contribuisce a una progettazione più modulare e manutenibile.
Relazione di dipendenza tra Utente e UtenteDAO è giustificata dal fatto che le operazioni di accesso al database coinvolgono oggetti Utente, ma i due elementi possono esistere e evolversi indipendentemente l'uno dall'altro.

quelle di utilizzo:
1. Relazione di Utilizzo tra DAO e Service:
- Scopo del DAO: La classe UtenteDAO ha il compito di interagire con il database, eseguendo query e operazioni specifiche per l'entità Utente. Queste operazioni coinvolgono la persistenza dei dati nel database.
- Scopo del Service: La classe ServiziUtenteImpl funge da strato intermedio tra il DAO e il Controller. Essa contiene la logica di business, che può includere la validazione dei dati, la manipolazione dei risultati del DAO e la gestione delle transazioni.
- Razionale per l'utilizzo: Il Service utilizza il DAO per accedere e manipolare i dati nel database. Ad esempio, il metodo login del ServiziUtenteImpl richiama il metodo login dell'UtenteDAO per ottenere un utente dal database.
- Vantaggi della Separazione: La separazione tra DAO e Service consente di mantenere la modularità e la chiarezza delle responsabilità nel sistema. Il DAO si concentra sulla persistenza dei dati, mentre il Service gestisce la logica di business.
2. Relazione di Utilizzo tra Service e Controller:
- Scopo del Service: Il Service contiene la logica di business e rappresenta un'interfaccia tra il DAO e il Controller. Esso implementa le operazioni di alto livello che il Controller richiama.
- Scopo del Controller: Il Controller gestisce le richieste provenienti dalla parte frontend dell'applicazione e coordina l'interazione tra il Service e la vista.
- Razionale per l'utilizzo: Il Controller utilizza il Service per eseguire operazioni di business logic in risposta alle richieste dell'utente. Ad esempio, il metodo loginIMP del UtenteController richiama il metodo login del ServiziUtenteImpl per gestire il processo di accesso.
- Vantaggi della Separazione: La separazione tra Service e Controller aiuta a mantenere il Controller focalizzato sul coordinamento delle richieste e delle risposte, mentre il Service si occupa delle logiche più complesse.
L'utilizzo di queste relazioni aiuta a strutturare il sistema in modo modulare, facilitando la manutenzione e l'estensione del codice nel tempo.

Relazione di dipendenza tra UtenteController e PianoController c'è perchè l'utentecontroller richiama un metodo del pianocontroller in un suo metodo