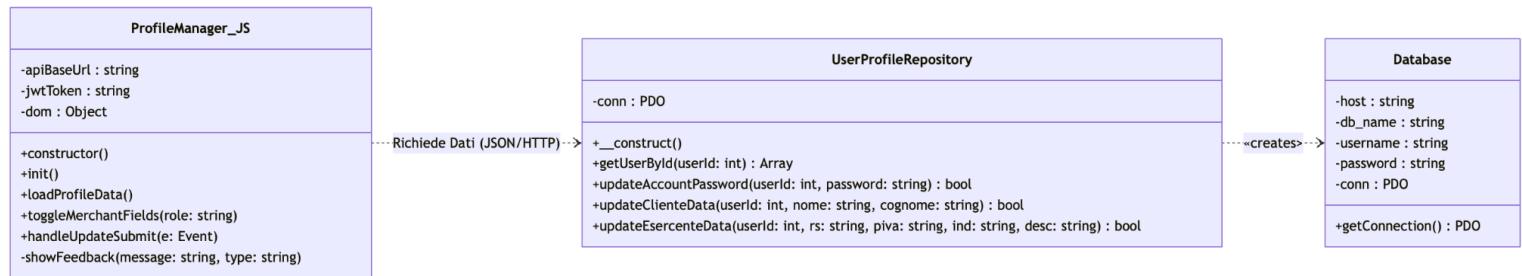


SVOLTO DA COLUCCI PASQUALE, MATR: 358141

Diagramma delle Classi: Modulo Gestione Profilo



Descrizione Architetturale delle Classi: Modulo Gestione Profilo

Il diagramma delle classi illustra l'architettura aggiornata del modulo, progettata per rispettare i requisiti di sicurezza (Stateless) e integrità dei dati. Il design mantiene l'approccio MVP (Model-View-Presenter), separando nettamente il client (browser) dal server (API).

1. Layer Presenter (Client-Side)

Classe: ProfileManager_JS

- Ruolo:** Agisce come Controller/Presenter lato client. Gestisce l'interazione con il DOM, l'invio asincrono dei dati e la gestione del Token di sessione.
- Attributi Chiave:**
 - jwtToken: Memorizza il token JWT recuperato dal LocalStorage. Sostituisce il vecchio ID utente in chiaro, garantendo che il client non manipoli l'identità dell'utente (RNF-2.1).
- Metodi Principali:**
 - init(): Verifica l'esistenza del token all'avvio (sicurezza client-side).
 - loadProfileData(): Effettua la chiamata GET iniettando l'header Authorization.
 - handleUpdateSubmit(): Gestisce la sottomissione del form. Esegue una validazione preliminare (es. match password) e costruisce il payload JSON per l'API.

2. Layer Model (Server-Side)

Classe: UserProfileRepository

- Ruolo:** Rappresenta il Model. Incapsula tutte le query SQL per la lettura e la scrittura dei dati dell'utente, astraendo la complessità del database dai controller API.
- Metodi di Aggiornamento:**
 - updateAccountPassword, updateClienteData, updateEsercenteData: Metodi granulari che eseguono query UPDATE

mirate sulle specifiche tabelle (utente, cliente_unipr, esercente), invocati dal controller solo dopo il superamento della validazione ridondante.

- **Gestione Risorse:** Nel costruttore `__construct()`, la classe istanzia un nuovo oggetto Database. Questo garantisce che ogni istanza del Repository abbia la propria connessione isolata.

Classe: Database

- **Ruolo:** Classe infrastrutturale per la configurazione e l'instaurazione della connessione PDO.
- **Scelta Progettuale:** La classe è progettata per essere istanziata on-demand. Non viene utilizzato il pattern Singleton. Questo approccio assicura che ogni richiesta HTTP gestita dai file PHP (`get_profile.php`, `update_profile.php`) operi su una connessione "fresca" e dedicata, evitando conflitti di stato e favorendo la scalabilità orizzontale.

3. Analisi delle Relazioni

Nel diagramma sono rappresentate due relazioni di Dipendenza chiave:

1. **ProfileManager_JS ..> UserProfileRepository:** Indica una dipendenza funzionale basata sul protocollo HTTP. Il Frontend non istanzia il Backend, ma ne consuma i servizi scambiando messaggi JSON. L'autenticazione di questa relazione è garantita dal **Token Bearer**.
2. **UserProfileRepository ..> Database («creates»):** Indica una relazione di dipendenza forte di tipo "Creazione". Il Repository è responsabile della creazione dell'istanza Database necessaria per il suo funzionamento. Poiché la connessione viene aperta e chiusa all'interno del ciclo di vita della singola richiesta (Stateless), non vi è persistenza di oggetti in memoria tra una chiamata e l'altra.