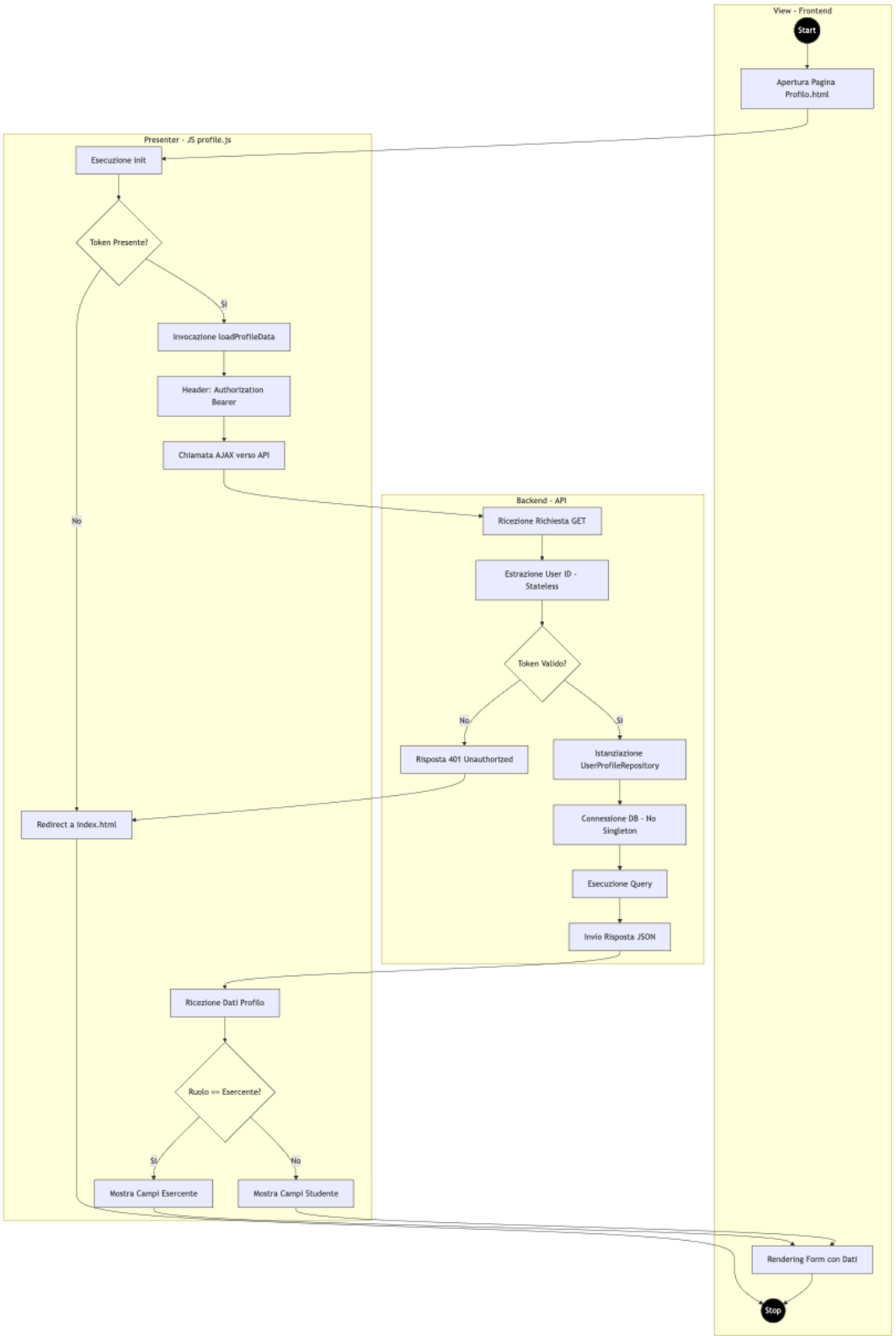
SVOLTO DA COLUCCI PASQUALE, MATR: 358141

Diagramma delle Attività: Visualizzazione Profilo, Modulo Gestione Profilo



Descrizione: Visualizzazione del Profilo, Modulo Gestione Profilo

Il diagramma delle attività modella il flusso operativo per il popolamento sicuro e dinamico dell'interfaccia utente. La logica è suddivisa in tre partizioni (Swimlanes) che separano la presentazione, la logica client e l'elaborazione server.

1. Partizioni (Swimlanes)

- **View (Frontend):** Rappresenta il layer di presentazione puro (DOM HTML `Profilo.html`). In questa fase è passivo e attende i dati.
- **Presenter (JS - profile.js):** Il controller lato client. Gestisce la logica di avvio (`init`), il controllo locale del token e le chiamate asincrone.
- **Backend (API - get_profile.php):** L'endpoint REST che funge da entry-point per la logica di business e l'accesso ai dati.

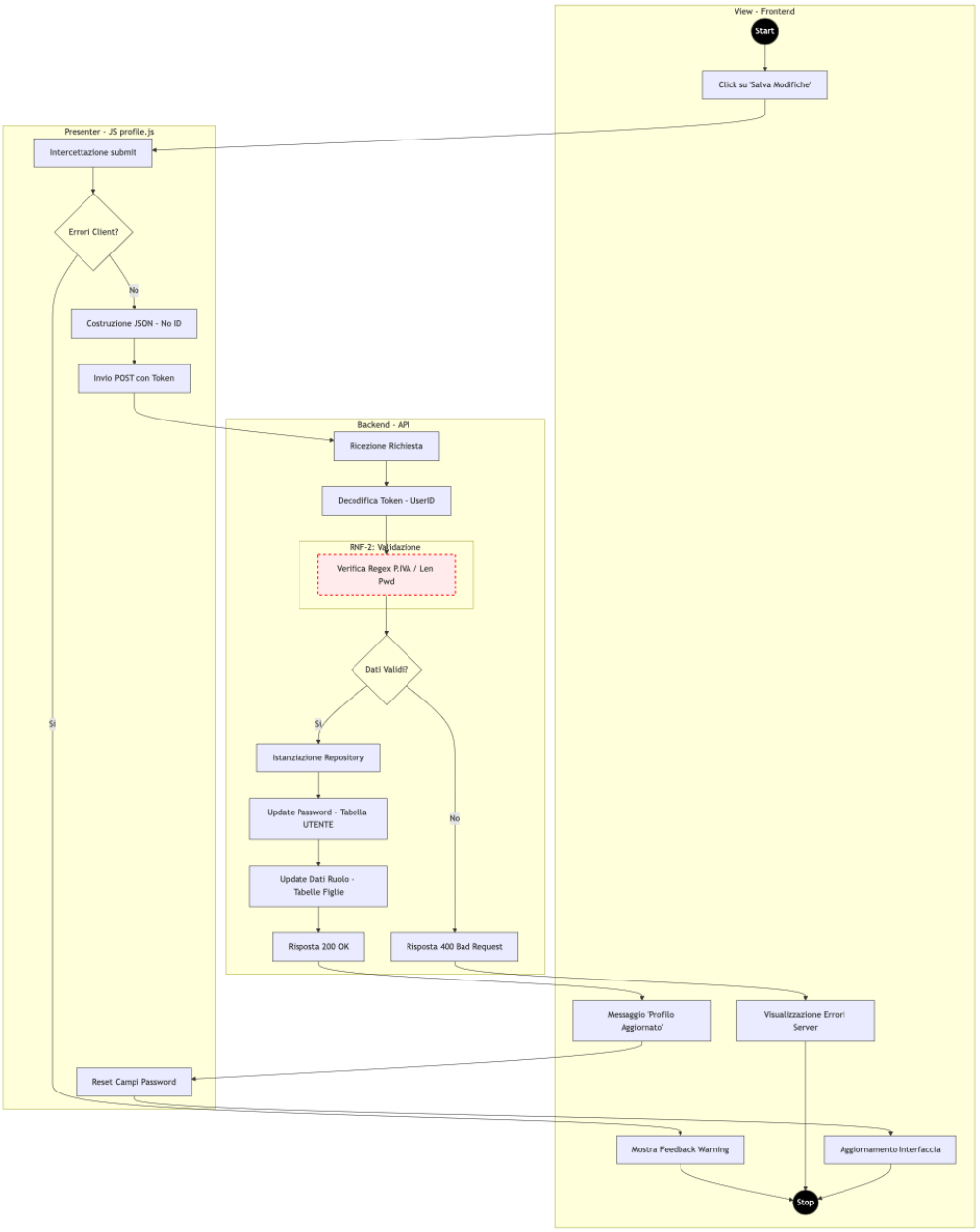
2. Flusso degli Eventi

1. **Avvio e Security Check:** All'apertura della pagina, il Presenter esegue immediatamente un controllo di sicurezza locale: verifica l'esistenza del token JWT nel `LocalStorage`.
    - *Ramo Token Assente:* Il flusso viene interrotto e l'utente reindirizzato al login (`index.html`), risparmiando una chiamata al server.
2. **Richiesta Stateless:** Se il token è presente, viene invocato `loadProfileData()`. La richiesta HTTP GET viene costruita inserendo il Token nell'header `Authorization`, senza passare ID in chiaro nell'URL.
3. **Elaborazione Backend:** Il server riceve la richiesta e:
    - Identifica l'utente decodificando il payload del Token (Stateless).
    - Se valido, istanzia puntualmente il `UserProfileRepository` e la connessione `Database` (non condivisa/No Singleton) per recuperare i dati.
4. **Adattamento Interfaccia:** Il Presenter riceve il JSON di risposta e, in base al valore del campo `ruolo`, adatta la View: mostra i campi P.IVA/Ragione Sociale se l'utente è un Esercente, o i campi Matricola se è uno Studente.

3. Note Tecniche

- **Autenticazione Stateless :** L'identità dell'utente non è dedotta da sessioni PHP, ma è calcolata matematicamente dal Token a ogni richiesta.
- **Gestione Risorse:** La connessione al Database viene aperta solo nel momento in cui la richiesta è autenticata e viene chiusa al termine dell'invio della risposta.

Diagramma delle Attività: Aggiornamento del Profilo, Modulo Gestione Profilo



Descrizione: Aggiornamento del Profilo, Modulo Gestione Profilo

Il diagramma illustra il processo di modifica dei dati utente, evidenziando l'implementazione del principio di **"Difesa in Profondità"** tramite la validazione ridondante (RNF-2).

1. Partizioni (Swimlanes)

- **View (Frontend):** L'interfaccia di input dove l'utente immette i nuovi dati.
- **Presenter (JS - profile.js):** Gestisce la pre-validazione (UX) e la costruzione del payload.

- **Backend (API - update_profile.php):** Esegue la validazione autoritativa di sicurezza e la persistenza dei dati.

2. Flusso degli Eventi

1. **Validazione Client (Primo Livello):** Al click su "Salva", il Presenter intercetta l'evento e verifica la coerenza formale (es. coincidenza password). Se fallisce, mostra un feedback immediato senza contattare il server.
2. **Invio Sicuro:** Se il client approva, viene inviato un payload JSON contenente solo i dati da modificare. L'ID utente **non** viene inviato, ma sarà estratto dal Token dal server.
3. **Validazione Ridondante (RNF-2):** Il Backend riceve la richiesta e, prima di qualsiasi operazione sul DB, esegue una **validazione rigorosa**:
    - Verifica Regex su P.IVA (11 cifre) e Nomi (no caratteri speciali).
    - Verifica lunghezza minima Password.
    - *Ramo Errore:* Se i dati sono malformati, risponde con `400 Bad Request`.
4. **Persistenza Transazionale:** Solo se la validazione server ha successo, viene istanziato il Repository per eseguire gli `UPDATE` sulle tabelle pertinenti (`Utente` per la password, `Cliente` o `Esercente` per i dettagli anagrafici).

3. Note Tecniche

- **Conformità:** Il sistema implementa una doppia barriera. Anche se un attaccante bypassa il controllo JavaScript (es. usando Postman), la validazione server-side blocca i dati invalidi proteggendo l'integrità del database.
- **Architettura No-Singleton:** Per garantire l'isolamento delle transazioni, ogni richiesta di aggiornamento opera su una nuova istanza della connessione al database.