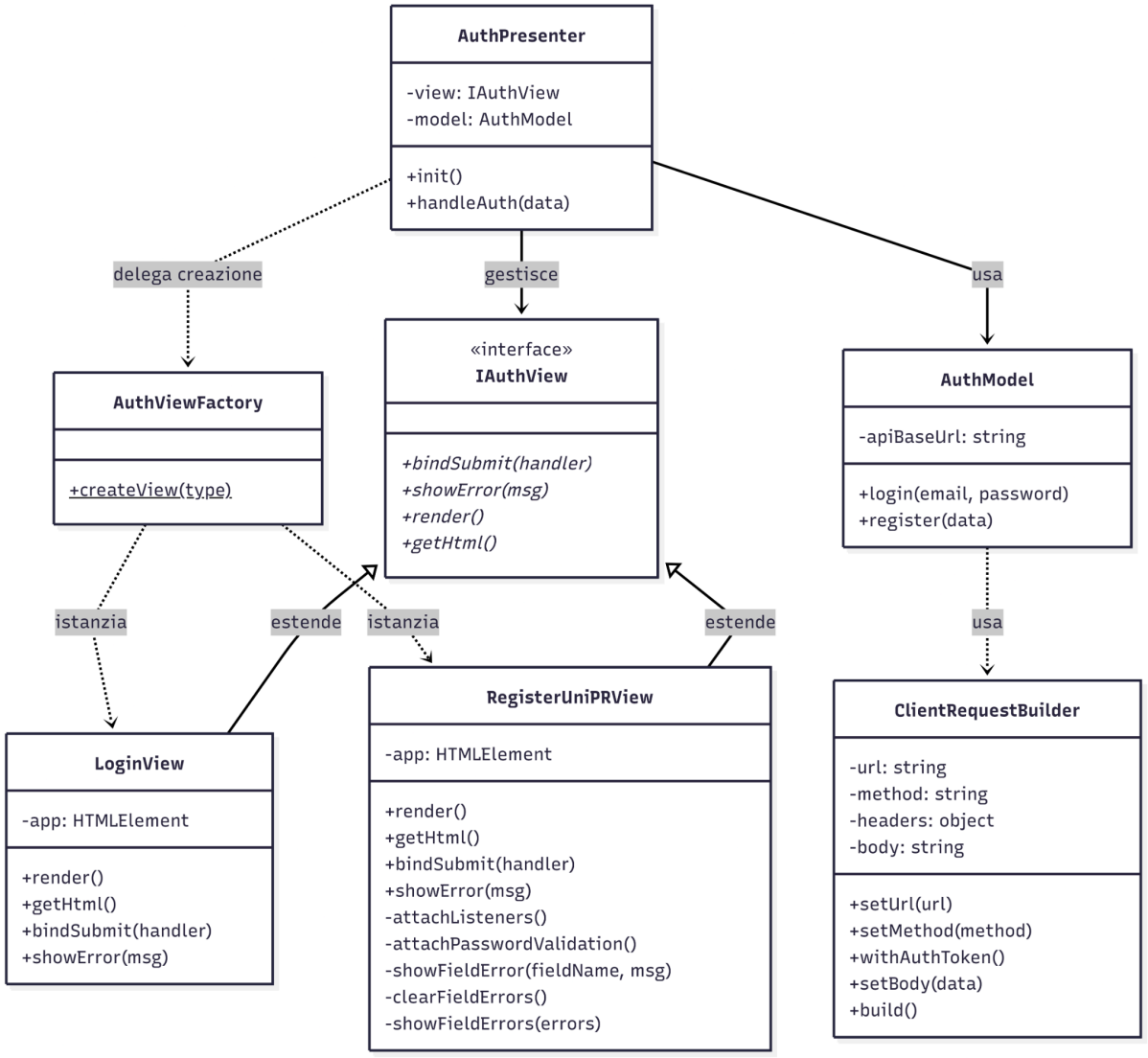
Svolto da: Sale Mario Matricola:364432

Diagramma delle classi:



1. Descrizione Architetturale

Il diagramma definisce la struttura statica del modulo di autenticazione, progettato secondo il pattern architetturale **MVP (Model-View-Presenter)** come richiesto dal vincolo RNF-1.2. Questa scelta garantisce il disaccoppiamento tra la logica di presentazione (View), la logica di controllo (Presenter) e l'accesso ai dati (Model). Le viste sono implementate come classi ES6 che ereditano da una classe base astratta, garantendo modularità e riutilizzo del codice.

2. Analisi del Layer Presenter e View

- **AuthPresenter:** È il componente centrale di orchestrazione lato client. Non interagisce direttamente con il DOM, ma comunica con le viste attraverso l'interfaccia astratta `IAuthView`. Gestisce la logica di business chiamando `bindSubmit` per reagire alle azioni dell'utente e coordinando il Model.
- **AuthViewFactory (Factory Method Pattern):** Implementa un pattern creazionale per centralizzare la logica di istanziazione delle viste. Tramite il metodo statico **`createView(type)`**, la Factory discrimina quale vista istanziare (`LoginView` o `RegisterUniPRView`) in base al parametro ricevuto, disaccoppiando il Presenter dalla creazione diretta degli oggetti grafici.
- **IAuthView (Interface):** Definisce il contratto che tutte le viste devono rispettare. Espone metodi astratti fondamentali come **`bindSubmit(handler)`** per il binding degli eventi e `showError(msg)` per il feedback utente, obbligando le classi concrete a implementarli.
- **RegisterUniPRView:** Questa classe concreta implementa la logica di presentazione per la registrazione. A differenza di una semplice vista statica, include metodi privati come **`attachListeners`** e **`attachPasswordValidation`** che implementano una logica **reattiva**: mostrano o nascondono dinamicamente il campo "Matricola" e validano la password in tempo reale manipolando il DOM, garantendo un feedback immediato (Pattern Observer sui campi input).

3. Analisi del Layer Model e Networking

- **AuthModel:** Gestisce la comunicazione asincrona con il Backend. Espone metodi ad alto livello (`login`, `register`) che restituiscono `Promise`, astraendo i dettagli del protocollo HTTP e della gestione degli endpoint (incluso il routing verso il gateway PHP).
- **ClientRequestBuilder (Builder Pattern):** Classe di utilità introdotta per gestire la complessità delle richieste API (Fetch API). Attraverso un'interfaccia fluente (`setUrl`, `setMethod`, `setBody`), permette di costruire richieste HTTP in modo leggibile e manutenibile.
- **Gestione Sicurezza:** Il metodo `withAuthToken()` predispone l'invio del Token JWT nell'header *Authorization* ove necessario, centralizzando la logica di autenticazione stateless e riducendo la ridondanza del codice nei vari servizi.