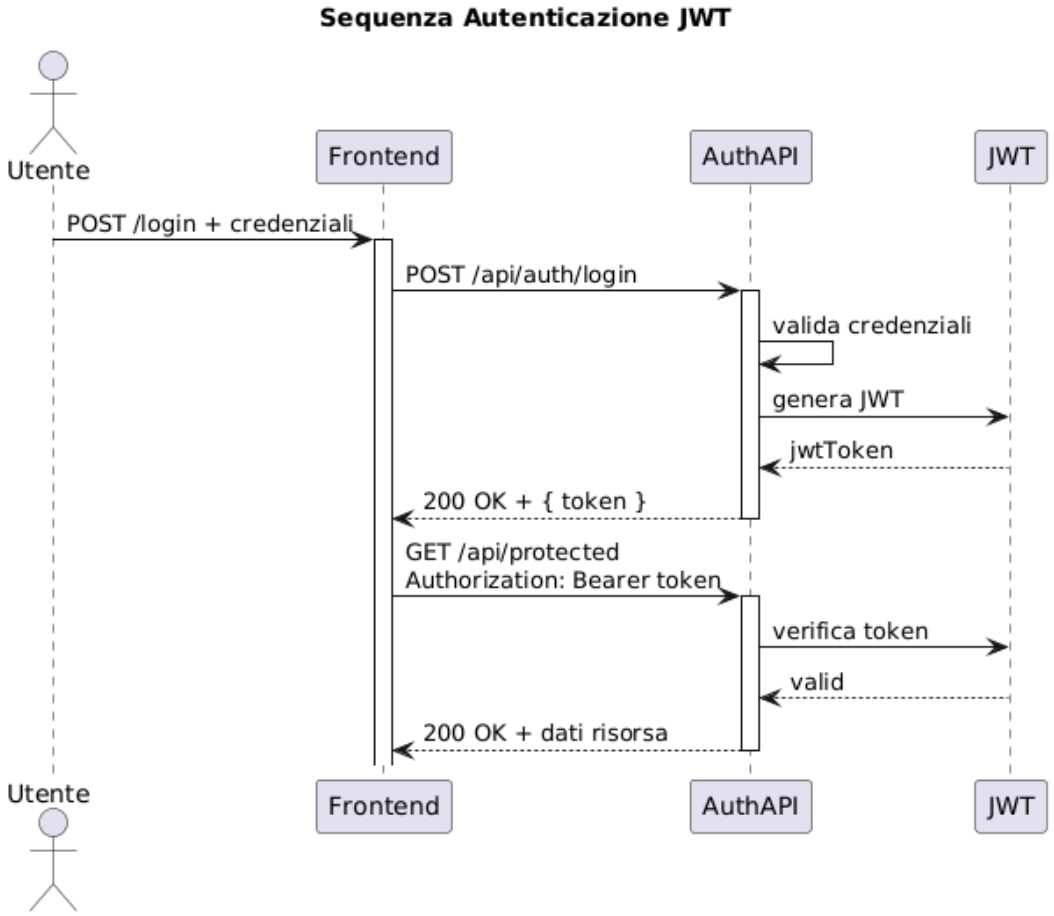
Sequence Diagram



Questo diagramma mostra il flusso di login, generazione token e uso dei JWT per proteggere le API:

Nel mio progetto

Nel `docker-compose.yml` ho `JWT_SECRET=supersegreto`. È la chiave usata da AuthAPI per firmare e verificare i JWT.

- **Backend**
  - `AuthController` espone `POST /api/auth/login` e `POST /api/auth/register`.
  - Alla ricezione delle credenziali, `UserService` interroga MySQL (`db` service) tramite il repository.
  - Se l'utente esiste e la password è corretta, il controller invoca un modulo JWT che crea il token.

- **Middleware JWT**
  - Tutte le rotte protette (es. `/api/corsi`, `/api/argomenti`) sono coperte da un middleware che:
    - Estrae l'header `Authorization`
    - Verifica firma e validità del token con lo stesso `JWT_SECRET`
    - Aggiunge `req.user = { id, ruolo }` per i controller successivi
- **Frontend**
  - Ho `REACT_APP_SERVER_URL=/api` al build: tutte le chiamate fetch/axios puntano automaticamente al mio backend su `http://localhost:3001/api`.
  - Dopo il login, React salva il token (localStorage /cookie) e lo include in ogni chiamata successiva:

    ```
    fetch('/api/protected', {
      headers: { Authorization: `Bearer ${token}` }
    })
    ```

    In questo modo grantisco che solo i client autenticati possano accedere ai dati.