



# Sistema di Gestione Presenze: Soluzione Centralizzata e Automatizzata

Il sistema offre una soluzione **centralizzata, affidabile e automatizzata** per la registrazione e gestione delle presenze del personale. Nasce per superare i limiti dei metodi manuali, **integrandosi con l'infrastruttura esistente** tramite **lettori RFID** per **monitorare ingressi e uscite in tempo reale**.

Il sistema **elimina gli errori manuali** e offre **dati in tempo reale**, aumentando l'**efficienza amministrativa**. Garantisce la **completa tracciabilità** degli accessi e, grazie a un'**architettura flessibile basata su API**, si integra facilmente con altri dispositivi e software gestionali.

# Utenti e Sistemi Target



## Sistemi di Lettura RFID

Client primari delle API che inviano dati ogni volta che un tag viene letto da un dipendente.



## Ufficio Risorse Umane

Utilizzatori delle interfacce dedicate per monitorare, correggere ed estrarre dati per l'elaborazione delle paghe.



## Manager/Supervisori

Accedono al sistema per verificare la presenza dei propri team e gestire le risorse in tempo reale.

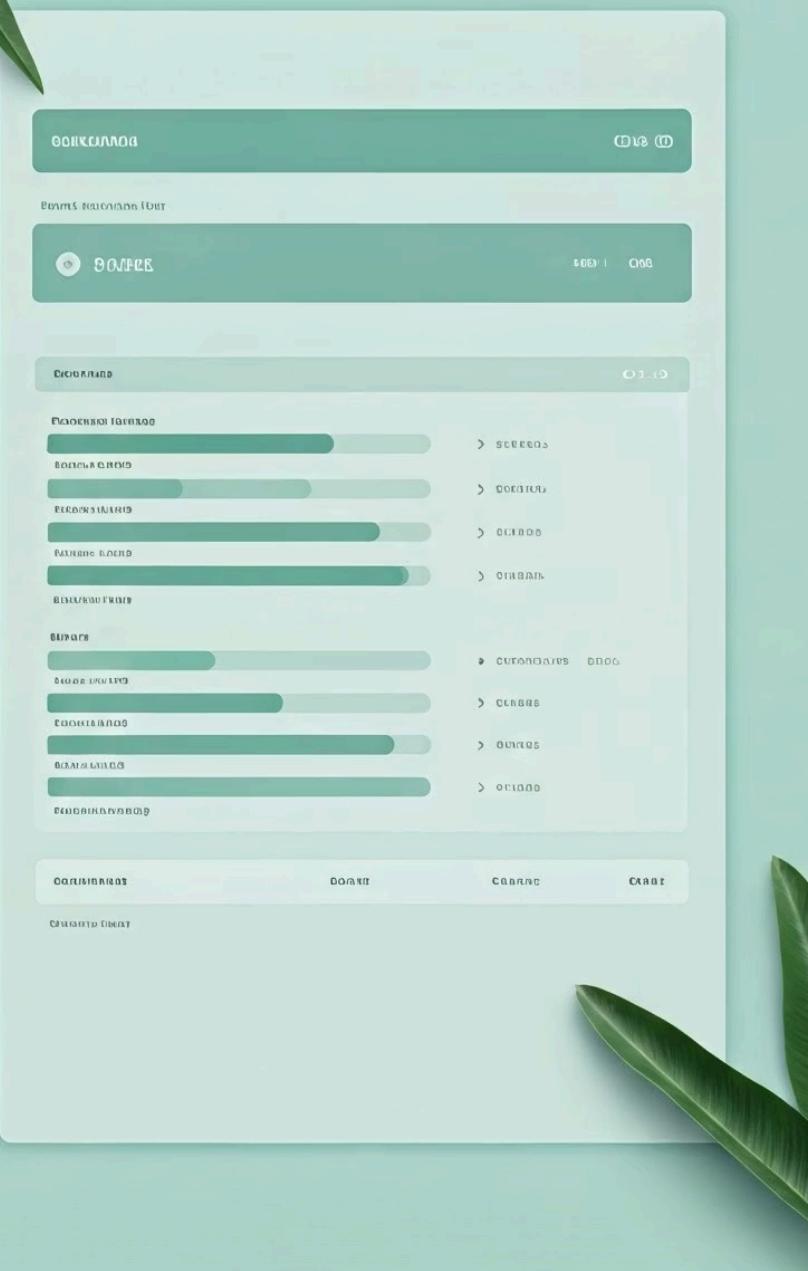


## Sviluppatori/Integratori

Professionisti che interfacciano nuovi lettori o sistemi esterni con questa piattaforma.



# Confini e Limitazioni del Sistema



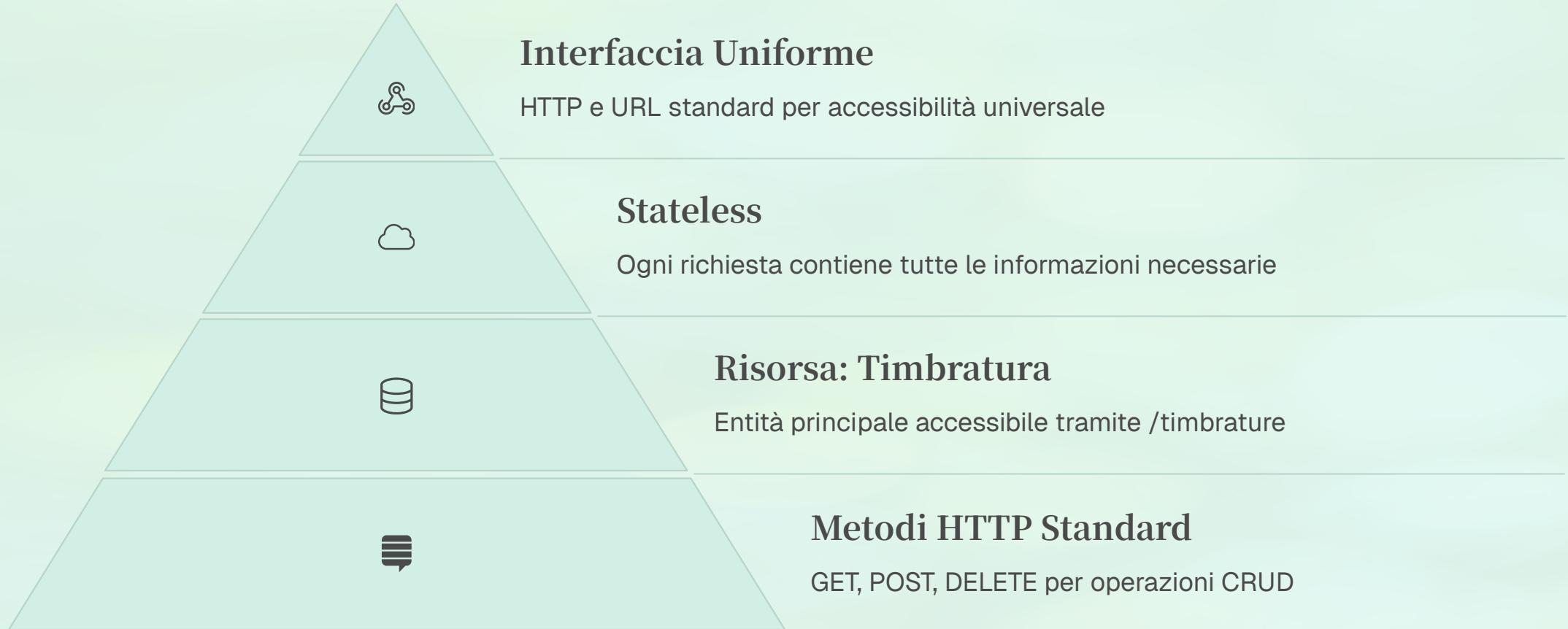
## Funzionalità Implementate

- Registrazione ingressi/uscite
- Validazione della sequenza
- Elenco delle timbrature odierne
- Eliminazione dell'ultima timbratura per utente

## Funzionalità Non Implementate

- Gestione anagrafica completa dei dipendenti via API
- Calcolo automatico delle ore lavorate/straordinari
- Gestione ferie/permessi
- Reportistica avanzata
- Autenticazione/autorizzazione API robusta

# Principi e Architettura API



# URL di Base e Versioning



## URL di Base

`http://://api`



## Versioning Attuale

Non implementato



## Raccomandazione Futura

Introdurre `/api/v1/timbrature`

Attualmente il sistema utilizza un URL di base semplice senza implementare un sistema di versioning delle API. Per garantire compatibilità e flessibilità nelle future evoluzioni del servizio, sarebbe consigliabile introdurre un sistema di versioning esplicito come `/api/v1/timbrature`.

# Flussi di Lavoro Tipici



## Uscita Standard

Mario (RFID0001) esce alle 17:30 (POST con  
rfid=RFID0001&totem=USCI0001 → 201 Created)



## Errore di Timbratura

Rientra e passa erroneamente davanti al lettore di uscita (409 Conflict: "Errore: Il dipendente è già registrato come USCITO")



## Correzione Ingresso

Passa davanti al lettore di ingresso (POST con  
rfid=RFID0001&totem=INGR0001 → 201 Created)



## Alternativa: Cancellazione

L'ufficio HR può cancellare l'ultima timbratura errata (DELETE con ?rfid=RFID0001 → 200 OK)

# Gestione degli Errori e Codici di Stato

## Successo

200 OK: Richiesta completata con successo (GET, DELETE)

201 Created: Risorsa creata con successo (POST)

## Documentazione

Il corpo della risposta (text/plain) contiene dettagli sull'errore



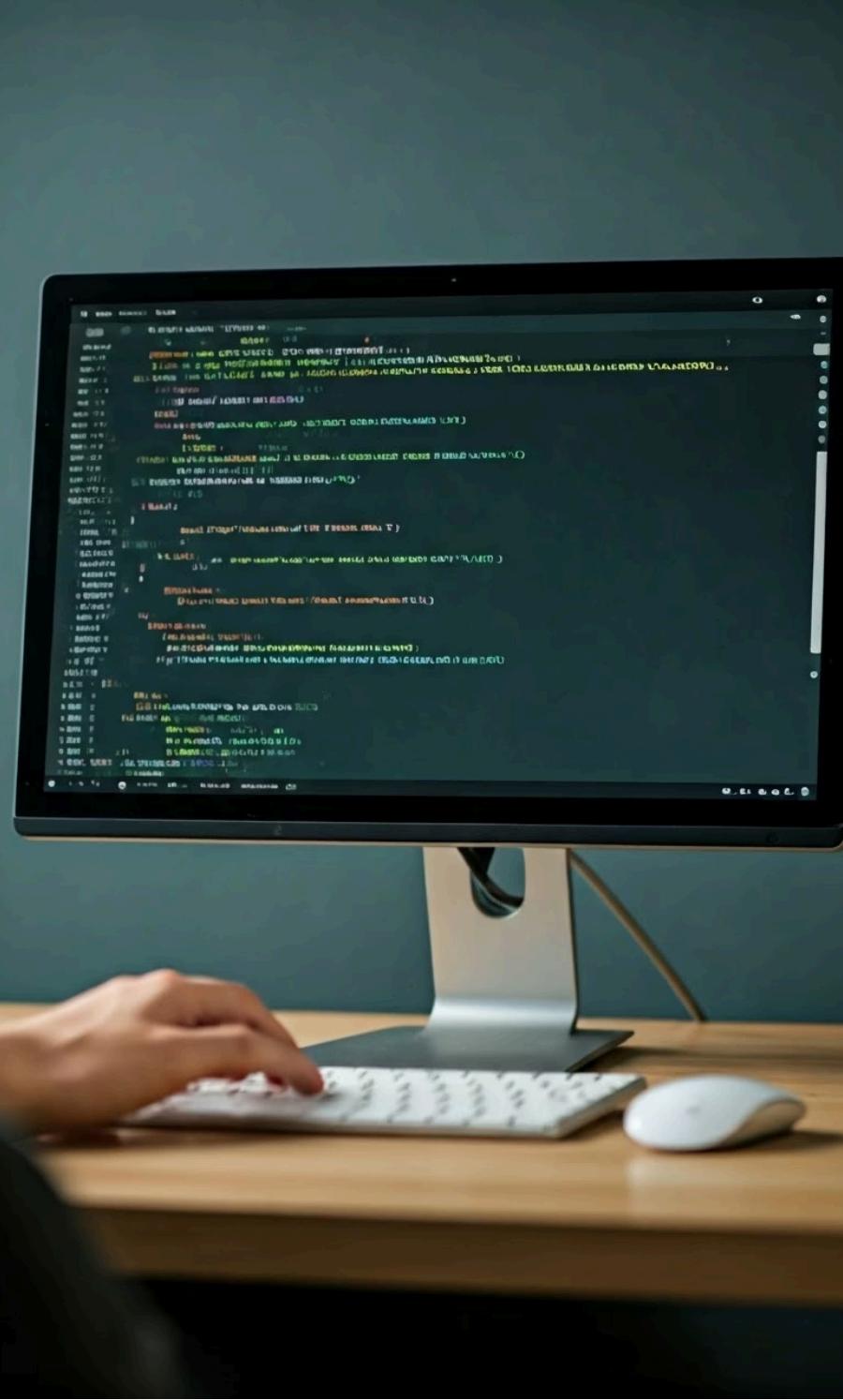
## Errore Client

400 Bad Request: Richiesta malformata

409 Conflict: Violazione regola di business

## Errore Server

500 Internal Server Error: Problema lato server



# Esempi di Interazione (curl)

## Recupero Timbrature

```
curl -X GET
```

```
http://localhost:8080/TimbraturaWebService/api/timbrature
```

## Registrazione Timbratura

```
curl -X POST -d "rfid=RFID0001&totem=INGR0001"
```

```
http://localhost:8080/TimbraturaWebService/api/timbrature
```

## Eliminazione Ultima Timbratura

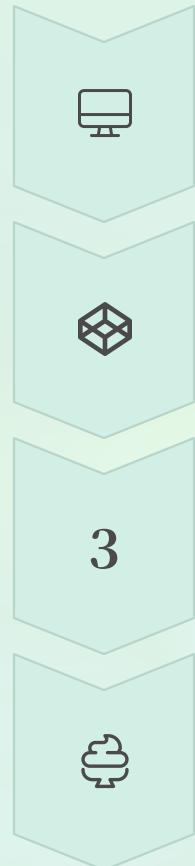
```
curl -X DELETE
```

```
http://localhost:8080/TimbraturaWebService/api/timbrature?  
rfid=RFID0001
```

Questi esempi mostrano come interagire con il sistema utilizzando comandi curl da terminale. Sono utili per testare il funzionamento delle API e possono servire come riferimento per gli sviluppatori che devono integrare il sistema con altre applicazioni.



# Architettura Software e Database



## Presentation/Client Layer

Interfaccia web (index.html) o sistemi esterni che interagiscono con le API

## Web/Controller Layer

TimbraturaServlet che gestisce le richieste HTTP e interpreta i parametri

## Data Access Layer

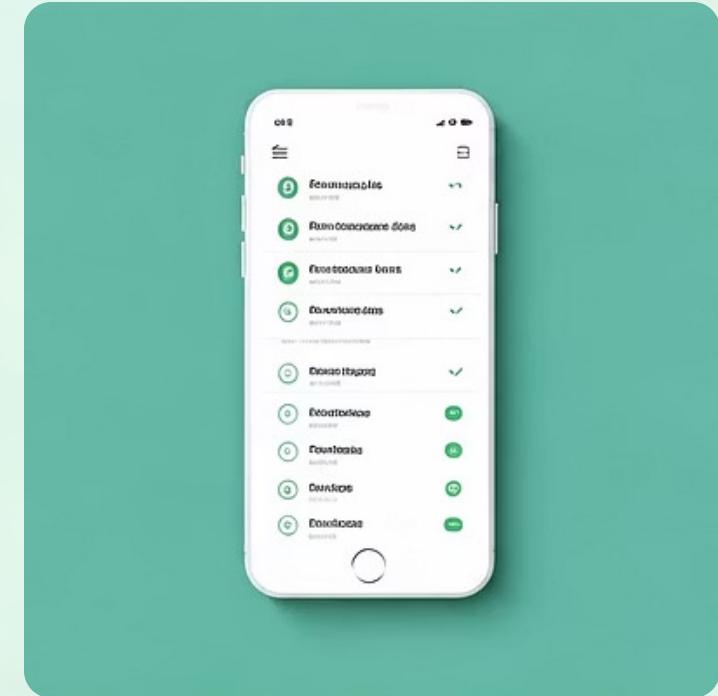
TimbraturaDAO che interagisce con il DB e implementa la logica di business

## Data Layer

Database MySQL con tabelle lavoratori, tornelli e timbrature

Il sistema segue un'architettura a layer semplificata, eseguita da Apache Tomcat. La struttura del database comprende tabelle per l'anagrafica base (lavoratori), la definizione dei lettori (tornelli) e gli eventi transazionali (timbrature).

# Sicurezza e Sviluppi Futuri



Attualmente il sistema non implementa misure di sicurezza, ma è essenziale aggiungere HTTPS, autenticazione (API Key/OAuth) e autorizzazione per proteggere i dati sensibili dei dipendenti.

Le possibili estensioni future includono: reportistica avanzata, gestione anagrafica via API, timbrature con causali, interfaccia web migliorata e sistema di notifiche. Questi sviluppi renderanno il sistema più completo e adatto alle esigenze aziendali in evoluzione.