# WEB-APP AZIENDALE

# Sistema di gestione delle prenotazioni di oggetti interni all’azienda.

****

#### Gruppo lavoro:

#### Falanga Domenico

#### Filadoro Antonio

#### Cappiello Antonino

#### Maddaloni Michele

#### Audino Luigi

#### Sessa Marco

Sommario

requisiti 4

Casi d’uso 4

analisi progetto5

Controlli effettuati5

# Requisiti

Lo scopo della nostra web-app è quella di gestire le prenotazioni di oggetti (sale riunioni, proiettori, sale conferenze), con tutti i controlli del caso, da parte degli utenti che lavorano in azienda. Gli utenti posso accedere al portale tramite le credenziali aziendali (email, password), in quanto si appoggia sul database aziendale in cui sono già registrati.

L’applicazione lato client gira su un web browser che attraverso opportune richieste HTTP comunica con il server. Per la realizzazione lato client (front end) è stato utilizzato il framework Angular 6 di Java, mentre per il lato server e DB (back end) è stato utilizzato il framework Spring Boot e PostgreSQL.

## Casi d’uso

Esistono due tipologie di attori:

* Admin;
* Utente;

Ed ognuno ha i suoi privilegi. L’utente puo’:

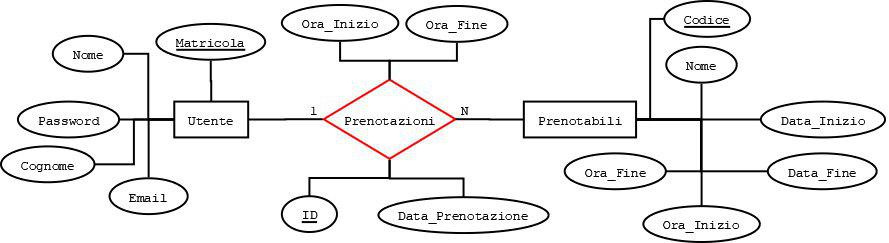
* Gestire il lato delle sue prenotazioni: inserimento, modifica e cancellazione;
* Gestire le proprie credenziali: profilo utente.
* Visionare tutti i prenotabili: ovvero gli oggetti che possono essere prenotati;

L’amministratore invece puo’:

* Gestire tutto il lato utente: inserimento, modifica e cancellazione;
* Gestire tutto il lato dei prenotabili: inserimento, modifica e cancellazione;
* Visualizzare tutte le prenotazioni effettuate dagli utenti;

All’ accesso viene effettuato un controllo sulle credenziali e reindirizzati al menù consono ai propri privilegi.

# Analisi progetto

Il database utilizzato è di tipo relazionale e presenta due entità e una associazione:

All’atto della normalizzazione del modello E-R sopra riportato, l’associazione “prenotazioni” viene tradotta in una entità con due vincoli di chiave esterna: “matricola” che riferisce l’entità “utente” e “codice” che riferisce l’entità “prenotabili”.

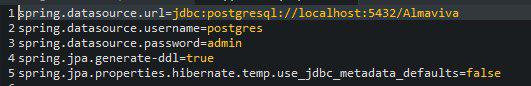
Le tabelle sono:

UTENTE = { Matricola, Nome, Cognome, E-mail, Password }

PRENOTAZIONI = { ID, (Matricola,Codice), Data, Ora\_inizio, Ora\_fine }

PRENOTABILI = { Codice, Nome, Data\_inizio, Data\_fine, Ora\_inizio, Ora\_fine }

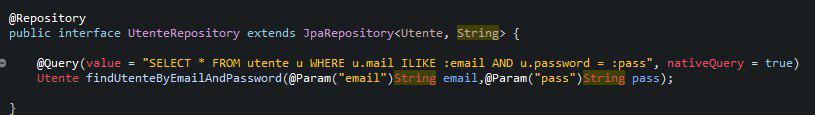
La realizzazione del back end è realizzata sfruttando il server Tomcat di Apache e la comunicazione invece avviene utilizzando i framework JPA e Spring Boot.



Nell’application properties andiamo ad inizializzare la connessione col database usando le righe di codice sopra riportata.

La realizzazione delle entità sul DB nel back end viene effettuata grazie al concetto di modello. Con le opportune “annotation” @Entity e @Table si associa la classe creata in Spring con la corrispondente entità nel DB. Ogni campo della classe deve essere annotato con @Column per indicare il nome della colonna del DB e nel caso della chiave primaria, si aggiunge anche l’annotazione @Id. Nelle classe “Prenotazioni” i vincoli di Foreign key vengono realizzati con l’annotazione @JoinColumn per effettuare la Join, e per la molteplicità invece si utilizza l’annotazione @OneToOne (nel nostro caso specifico).

Per l’effettiva comunicazione tra front-end e back-end, abbiamo utilizzato uno strato “intermedio” tra i modelli JSON ricevuti dal client con quelli effettivi realizzati in Spring. Tale strato intermedio viene implementato con le classi “ViewModel”. Tali classi sfruttano un’interfaccia “converter” che implementa metodi per la conversione da Model -> ViewModel e viceversa.



Le interrogazioni sul DB da parte degli utenti vengono effettuate utilizzando le interfacce Repository che estendono la classe astratta “JpaRepository” che mette a disposizione interrogazioni base come la ricerca o la cancellazione sulla base della chiave primaria, o ricerca e cancellazione totali. Tuttavia, queste interfacce offrono anche la possibilità di “customizzare” query con una specifica notazione (@Query) e opportune parametrizzazioni (@Param).

Le repository vengono richiamate con l’annotazione @Autowired all’interno dei vari Service: questi offrono dei servizi all’utente, eseguendo le operazioni specifiche.7

I servizi vengono lanciati dai Controller: questi ricevono dal client delle richieste HTTP utilizzando gli URI, in cui si possono specificare anche dei parametri di ingresso. I parametri di ingresso passati dagli URI vengono parametrizzati sfruttando l’annotazione @PathVariable mentre gli oggetti veri e propri, scambiati come file .JSON, vengono richiamati grazie all’annotazione @RequestBody. Inoltre tutti i metodi implementati in queste classi presentazione annotazioni del tipo @RequestMapping che si specializzano in:

* @GetMapping: per ricevere dal DB;
* @PostMapping: per inserire nel DB;
* @PutMapping: per modificare nel DB;
* @DeleteMapping: per cancellare nel DB;

Tutti i metodi ritornano file .JSON al front-end.

## Controlli effettuati

* **Controllo Login**: uno dei più importanti riguardo l’accesso alla web-app, che viene realizzato controllando se le due stringhe inserite (email, password) identificano un utente presente nel DB: a seconda dell’esito, ci fa effettuare o meno l’accesso.
* **Controllo inserimento utente/prenotabile**: per entrambe le entità, quando viene effettuato un nuovo POST, c’è il controllo se si viola il vincolo di chiave primaria: ovvero se si cerca di inserire una nuova tupla con una chiave primaria già presente nel DB. Inoltre, per il “Prenotabile” controlla anche se la Data\_inizio e l’Ora\_inizio sono effettivamente antecedenti rispettviamente Data\_fine e Ora\_fine.
* **Controllo modifica prenotabile**: quando si effettua la modifica di un prenotabile (possibile solo dall’admin), c’è un controllo sulle date e sull’orario: non posso inserire una data o un orario di inizio maggiore di quello di fine.
* **Controllo inserimento prenotazione**: quando un utente effettua una prenotazione, scattano diversi controlli, analizzati di seguito:
  + **Controllo date/orario sovrapposte**: controlla se la prenotazione che si sta inserendo non vada a sovrapporre date e/o orari in cui sono già presenti delle prenotazioni per lo stesso prenotabile.
  + **Controllo disponibilità prenotabile**: i prenotabili hanno degli intervalli di date e orari in cui sono disponibili: all’atto della prenotazione, si verifica che tali intervalli siano rispettati.