

# Relazione progetto di BASE DI DATI

**WEB APP** 

Prof. Stefano Paraboschi

SALVETTI DAVIDE MAT. 1057596 VERZEROLI MATTEO MAT. 1057926

## RELAZIONE PROGETTO BASI DATI

#### Descrizione del problema

Il progetto si prefigge come obiettivo quello di poter essere utilizzato come supporto all'organizzazione di un Centro Ricreativo Estivo.

All'interno dell'organizzazione sono presenti diverse figure, ognuna delle quali ha compiti ben precisi. Ogni figura potrà accedere all'applicazione grazie a una matricola e una password, le quali permetteranno di identificarne il ruolo e indirizzarli verso le relative homepage. La base di dati permetterà inoltre di salvare alcune informazioni anagrafiche di tutti i partecipanti: nome, cognome, data di nascita, email, indirizzo, numero di telefono, cellulare (non obbligatorio) e nominativo dei genitori (nel caso dei bambini).

A capo dell'organizzazione troviamo la figura di uno (o eventualmente di più) *leader*, i quali hanno la possibilità di svolgere diverse azioni sulla base dati. Sono infatti autorizzati a registrare gli altri partecipanti: bambini, animatori, responsabili, segretarie ed esterni. Non è possibile, attraverso la homepage, inserire altri leader, i quali verranno inseriti solo programmaticamente nella base dati. Un'altra operazione che il leader può effettuare è la creazione delle *Squadre*, indicandone il *nome*, *colore associato e motto*. Ogni bambino e animatore, infatti, sarà associato a una squadra la quale potrà partecipare a diversi eventi (*gite*, *laboratori* o *giochi*), anche essi definiti dai leader.

Le *git*e, come tutti gli eventi, sono identificati da un *id*, da una *descrizione*, dal *luogo*, dalla *data*, e dall'*ora* in cui si svolgono. Inoltre, il leader che organizza la gita dovrà indicare le squadre che possono parteciparvi. Inoltre, un bambino, appartenente a una squadra a cui è permessa la partecipazione alla gita, sarà in grado di iscriversi, indicando anche la data di iscrizione. L'animatore potrà quindi visualizzare l'elenco dei partecipanti a una particolare gita.

I *laboratori* sono inoltre associati a un esterno, ossia una persona non direttamente legata all'organizzazione, che potrà essere inserita all'interno della base dati solamente se associata a un laboratorio. Inoltre, alla creazione del laboratorio, sarà quindi possibile indicare l'esterno che avrà il compito di gestirlo.

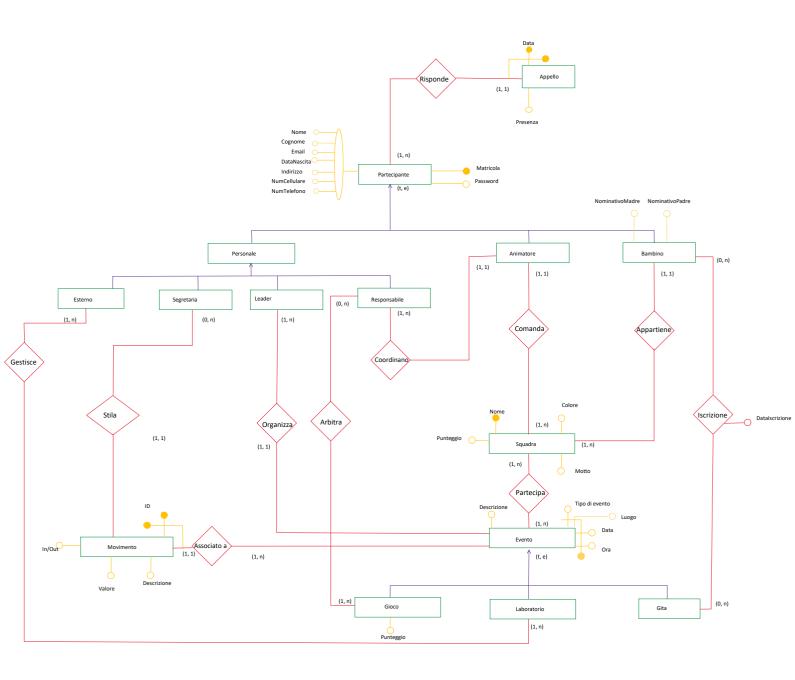
I *giochi* sono definiti anche da un punteggio (da 1 a 5) che sarà assegnato alla squadra da un responsabile nel caso di vittoria. Ai responsabili è inoltre possibile visualizzare la classifica generale di tutte le squadre, in base al loro punteggio totale. Ad ogni gioco è anche associato un arbitro, rappresentato da un responsabile, e assegnato dal leader al momento della creazione del gioco.

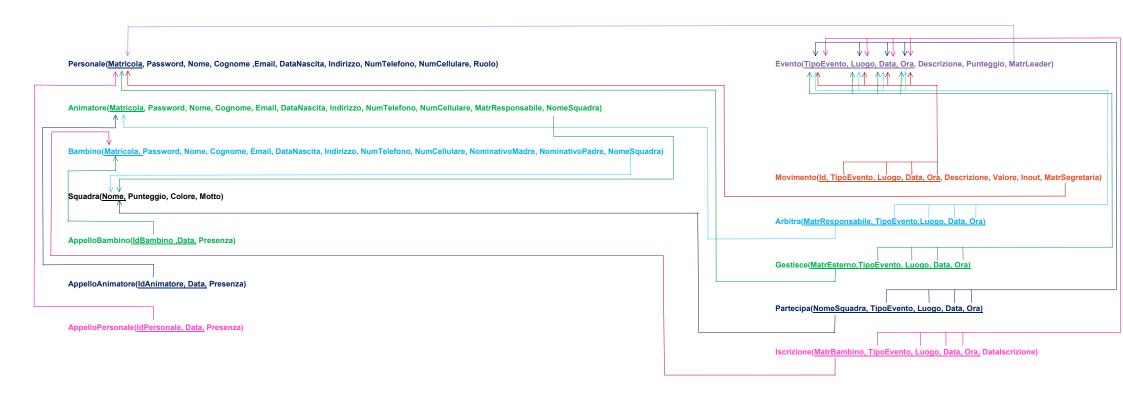
Tutto il personale, che comprende le figure dei leader, delle segretarie, dei responsabili e degli esterni, ha la possibilità di compilare un proprio appello, per indicare la loro presenza in un determinato giorno. Solo il leader potrà poi vedere un elenco delle presenze in un determinato giorno. I responsabili, invece, saranno in grado di compilare, sempre per un particolare giorno, le presenze degli animatori da loro coordinati, e visualizzarne l'elenco delle presenze (relativo a un particolare giorno). Infatti, il leader, quando crea un animatore, deve specificare il responsabile a cui deve fare riferimento. Infine, gli animatori potranno compilare l'appello per i bambini della loro squadra e visualizzare l'elenco relativo a un particolare giorno.

Tutti i partecipanti, nella loro home, potranno poi vedere un elenco degli eventi inseriti a calendario. In particolare, agli esterni saranno visualizzati solamente gli eventi a cui devono presiedere. Ai responsabili saranno inoltre resi visibili i giochi che sono chiamati ad arbitrare e agli animatori e ai bambini saranno resi visibili solo gli eventi relativi alla propria squadra.

Le segretarie hanno inoltre la possibilità di inserire dei movimenti legati agli eventi, indicando una descrizione e un importo (positivo per rappresentare un'entrata, negativo per rappresentare un

costo). Inoltre, gli sarà possibile visualizzare un elenco di tutti i movimenti effettuati da tutte le segretarie. Come tutti i partecipanti, saranno in grado, nella loro homepage, di modificare i propri dati (eccetto il numero di matricola), rispettando i formati richiesti e inoltre, a differenza del leader, tutti i partecipanti potranno eliminarsi dalla base di dati (problema che sarà discusso più avanti). Tutti i partecipanti, ad eccezione degli esterni, saranno in grado di visualizzare una tabella riepilogativa del numero di partecipanti iscritti al cre.





#### Sviluppo dell'app

Per lo sviluppo della nostra Web App abbiamo scelto di utilizzare lato *backend* il server web http integrato nella libreria Flask, scritta in Python, definendo poi le funzioni che devono essere eseguite ad ogni particolare richiesta client. Lato *frontend* abbiamo scelto di utilizzare pagine web dinamiche, scritte in HTML5 e CSS3, utilizzando le funzionalità offerte da Bootstrap. Grazie alle funzionalità offerte da HTML5 siamo stati in grado di integrare all'interno delle pagine tutti i controlli necessari sui dati inseriti dall'utente, in modo da mantenerne l'integrità con le specifiche stabilite. Ad esempio, abbiamo deciso che la password avrebbe dovuto essere composta da 5 caratteri (numeri compresi). Il controllo sul rispetto di tale regola è interamente delegato al client, il quale prima di inviare una richiesta effettua tutti i controlli sui dati inseriti. Altri esempi sono i controlli sull'inserimento di dati '*required*', ossia su dati che devono essere necessariamente inseriti dall'utente e senza i quali non viene inviata alcuna richiesta al server. Infine, è stato possibile anche visualizzare un calendario e un orologio per introdurre campi data e ora, e abbiamo potuto definire campi *number*, i quali permettono l'inserimento di soli numeri. Grazie a Bootstrap, una libreria che contiene modelli di progettazione basati su HTML e CSS, abbiamo basato la nostra web app sul 'Dashboard template', utilizzando anche alcuni metodi di input e form disponibili sul sito.

Un altro problema che abbiamo dovuto affrontare, che caratterizza anche il protocollo http, è il mantenimento della sessione utente. Infatti, il protocollo http è session-less, ossia non permette direttamente di mantenere traccia di una sessione tra client e server. Grazie alle funzionalità messe a disposizione da Flask, abbiamo potuto risolvere questo problema utilizzando un approccio clientside session, dove i dati della sessione vengono memorizzati lato client. In particolare, Flask implementa l'utilizzo dei signed cookies, che permettono il salvataggio dei dati di sessione del client tramite l'utilizzo dell'oggetto session (implementato come un dizionario, in grado anche di mantenere traccia delle modifiche). Infatti, un noto metodo di risoluzione del problema della sessione in ambito web è quello di memorizzare informazioni nei cookies. Tuttavia, se i dati fossero memorizzati senza prevederne una criptazione, essi sarebbero chiaramente vulnerabili, in quanto visibili e modificabili da chiunque. Per questo motivo, Flask utilizza una chiave di caratteri (che nel nostro caso è generata casualmente dal sistema operativo all'avvio del server) per crittografare i contenuti dei cookies memorizzati. In questo modo, solamente chi è a conoscenza della chiave è in grado di poterli modificare. Inoltre, i signed cookies permettono anche di verificare se essi sono stati modificati. Nel caso in cui il server non riuscisse a decodificare i dati nel modo corretto scarterebbe i dati e cancellerebbe il loro contenuto, inizializzando una nuova session. Utilizzando quindi il concetto di sessione, siamo stati inoltre in grado di gestire un requisito fondamentale: permettere l'accesso a particolari pagine solo a chi avesse effettuato il login e avesse avuto l'autorizzazione per farlo.

Per quanto riguarda l'implementazione della base dati, abbiamo scelto di utilizzare il DBMS SQLite, utilizzando la libreria di Python sqlite3. Per rispondere ai requisiti minimi di sicurezza (ed evitare *SQL injection*), abbiamo utilizzato *statement parametrizzati*, utilizzando il *placeholder* offerto dalla libreria ('?').

Un altro problema che abbiamo dovuto affrontare è stata la gestione delle cancellazioni dei partecipanti al CRE. Infatti, un partecipante poteva essere legato tramite chiavi esterne a diversi vicoli. Tuttavia, non ci sembrava corretto applicare un metodo di eliminazione a cascata, il quale avrebbe portato a una eccessiva cancellazione di dati pregiudicando l'integrità della base dati stessa. Abbiamo quindi preferito che tutti i dati inseriti da utenti che in seguito si sono cancellati, ad esempio gli appelli, le iscrizioni agli eventi o i movimenti creati, fossero rimasti nel database in modo da poterli recuperare. Quindi abbiamo scelto di introdurre un elemento "fantasma" nelle table PERSONALE, ANIMATORE e BAMBINO in modo tale da riferirsi a loro per risolvere le violazioni dei vincoli di chiave esterna nel caso di cancellazione. Così facendo inoltre è possibile stabilire se una particolare tupla era associata a chiavi esterne che sono poi state eliminate.

### Pagine HTML

