UNIVERSITÀ DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA
CORSO DI LAUREA TRIENNALE IN
INGEGNERIA INFORMATICA

LudoTech

Documentazione Tecnica e Funzionale
Programmazione Web e Mobile

Gabriele Palma, 0751854

Lorenzo Cumella, 0763096

Valeria Bosco, 0764569

ANNO ACCADEMICO 2024 - 2025

Indice

1 Introduzione all'applicazione

LudoTech è una piattaforma web progettata per promuovere, diffondere e condividere esperienze di gioco digitale, attraverso la diffusione di minigiochi. L'applicazione nasce dall'idea di creare un ambiente versatile e accessibile, in grado di soddisfare sia gli utenti alla ricerca di un'esperienza ludica immediata, sia gli sviluppatori desiderosi di proporre e condividere le proprie creazioni.

La piattaforma persegue un duplice obiettivo: da una parte, mettere a disposizione degli utenti una selezione sempre aggiornata di minigiochi già pronti, che possono essere avviati e giocati direttamente dal browser senza necessità di download o installazioni; dall'altra, offrire un'opportunità concreta di visibilità a chi sviluppa giochi, permettendo il caricamento e la pubblicazione di minigiochi realizzati autonomamente.

1.1 Scopo del documento

Questo documento descrive l'architettura software, le tecnologie utilizzate e le scelte di implementazione alla base della piattaforma LudoTech. L'obiettivo è fornire una panoramica tecnica completa e giustificare le scelte di implementazione adottate nel corso dello sviluppo del progetto.

2 Architettura Generale e Stack Tecnologico

In questa sezione viene descritta l'architettura di alto livello dell'applicazione e le principali tecnologie che compongono lo stack.

2.1 Frontend

- **Framework/Libreria:** Angular 19 con il toolkit UI Ionic
- **Linguaggi:** HTML5, CSS3, JavaScript, TypeScript
- **Gestione dello Stato:** Approccio stateless, ogni componente recupera i dati necessari tramite opportuni endpoint nel backend
- **Routing:** Angular Router
- **Comunicazione HTTP:** Angular HttpClientModule

2.2 Backend

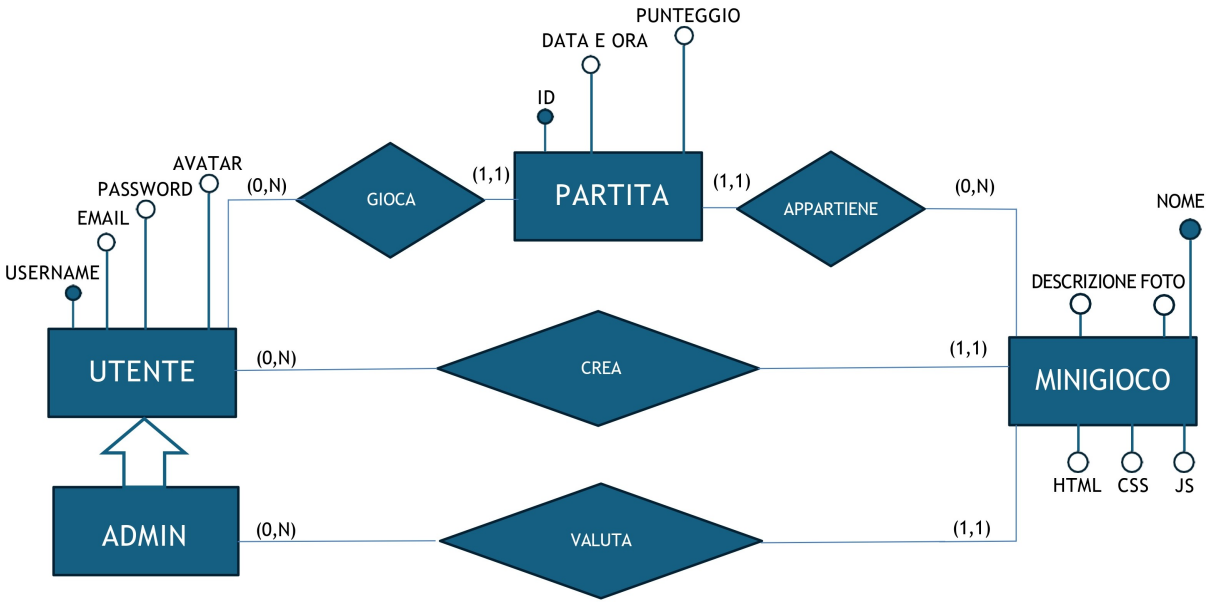
- **Framework:** Node.js con Express
- **Linguaggio:** JavaScript
- **Autenticazione:** gestione delle sessioni con cookie HTTP only
- **Interazione con DB:** Diretta tramite modulo Node.js sqlite3

2.3 Avvio in ambiente di sviluppo locale

- **Hosting locale Frontend:** nel terminale, `ng serve` nella cartella "ionic_frontend"
- **Hosting locale Backend:** nel terminale, `node server.js` nella cartella "express_backend"

2.4 Database

- **DBMS:** Sqlite3
- **Schema ER:**



3 Analisi delle Funzionalità e Dettagli Implementativi

3.1 Gestione Utenti (Registrazione, Login e Profilo)

L'applicazione gestisce l'identità degli utenti attraverso un sistema di registrazione e login. Una volta autenticato, l'utente può accedere a una pagina personale per visualizzare e modificare le proprie informazioni (nickname, email, avatar) o effettuare il logout.

3.1.1 Dettagli Implementativi

- **Registrazione:** L'utente compila i campi delle credenziali "username", "email" e "password" e le invia tramite richiesta `POST` all'endpoint di registrazione (1) del backend, che le riceve e le valida. In caso di errori, il backend manda una risposta al frontend contenente un messaggio di errore personalizzato, altrimenti carica l'utente nel Database e manda un messaggio di successo.
- **Registrazione utenti amministratori:** Per ragioni di sicurezza, non è possibile creare utenti amministratori tramite richieste come per gli altri utenti. Invece, all'avvio del server, esso tenterà automaticamente di inserire nel database un utente amministratore, se non esiste già, con variabili di ambiente come credenziali.
- **Login:** Le credenziali "username" e "password" vengono inviate tramite una richiesta all'endpoint di login (2). Il backend verifica le credenziali, cercando se esiste un utente col nome fornito e confrontando l'hash della password. In caso di successo, viene creata una sessione per l'utente e il frontend riceve una risposta affermativa contenente i dati del profilo, procedendo al reindirizzamento verso la Dashboard.

- **Sicurezza Password:** Le password vengono salvate all'interno del database al momento del caricamento dell'utente tramite hashing bcrypt. Al momento del login la password inviata dall'utente viene a sua volta criptata e confrontata con l'hash della password salvato tramite il metodo `bcrypt.compare()` .

- **Aggiornamento informazioni utente:** Attraverso la propria area utente personale, ogni utente potrà modificare le proprie informazioni utente, quali avatar (scegliendone uno tra gli avatar pre impostati), nome utente e email. Una volta terminata la modifica, l'utente invierà le proprie informazioni aggiornate al backend tramite apposito endpoint (5), che le controllerà e risponderà con l'esito della richiesta.

- **Logout utente:** Attraverso la propria area utente personale, l'utente loggato potrà inoltre effetturare la procedura di logout attraverso il bottone logout, che manderà una richiesta all'apposito endpoint (6). Il backend controlla la richiesta e, se è valida, elimina la sessione dell'utente e risponde con un messaggio di successo. L'utente a questo punto viene reindirizzato alla pagina di login.

- **API Endpoints:**

  1) `POST /api/auth/register` Registra un nuovo utente.
  2) `POST /api/auth/login` Autentica un utente e restituisce le sue informazioni.
  3) `GET /api/auth/me` Recupera i dati dell'utente autenticato.
  4) `GET /api/auth/users/:id` Recupera le informazioni di un utente in base al suo id.
  5) `PUT /api/auth/me` Aggiorna le informazioni del profilo.
  6) `POST /api/auth/logout` Logout utente attualmente loggato.

3.2 Dashboard e Fruizione Giochi

Dopo il login, l'utente viene reindirizzato alla Dashboard, che mostra l'elenco dei giochi disponibili e, se l'utente loggato è un amministratore, l'elenco dei giochi in attesa di approvazione. Ogni gioco è rappresentato da un box cliccabile con nome e icona. Cliccando, si accede a una pagina dedicata, divisa in diverse sezioni:

- **Gioco:** contiene la schermata di gioco, attraverso la quale l'utente potrà giocare tramite mouse e tastiera, in base all'implementazione del gioco in sè,

- **Info:** contiene l'icona del gioco, il suo nome, la descrizione e un ulteriore riquadro con le informazioni dell'autore del gioco,

- **Leaderboard:** contiene una tabella con i 10 migliori punteggi attualmente presenti per il gioco in questione (se presenti), seguita da un tasto per aggiornare i punteggi.

- **Admin:** sezione dedicata alle Azioni Amministrative, contiene un bottone "scarica file", un bottone "cancella gioco" e un bottone "approva gioco" se il gioco in questione non è ancora stato approvato da un amministratore.

3.2.1 Dettagli Implementativi

- **Recupero dati:** Quando l'utente richiede la pagina Dashboard, la lista dei giochi approvati viene recuperata tramite una richiesta all'endpoint (7). Se l'utente loggato è un amministratore, il sito tenterà inoltre di caricare i giochi non approvati (ovvero

i giochi dove il campo `approvedBy` è uguale a `null`), tramite richiesta all'endpoint (8). Una volta cliccata la box del gioco che si intende caricare il sito recupererà tutte le informazioni su di esso e sul suo autore tramite gli appositi endpoint (9 e 4).

- **Rendering del gioco:** La schermata di gioco assegna ad un iframe l'URL del file HTML del gioco in questione, servito staticamente dal backend nella cartella /uploads/<nome gioco> assieme ai relativi file CSS e JS a cui il file HTML farà riferimento.

- **Cancellazione del gioco:** All'interno della sezione amministrativa, l'admin avrà la possibilità, tramite apposito endpoint (10), cancellare il gioco in questione, causandone l'eliminazione dei file nel backend e della relativa riga nella tabella games, con conseguente eliminazione dei punteggi associati a quel gioco.

- **API Endpoints:**

  7) `GET /api/games/` Restituisce la lista dei giochi approvati.
  8) `GET /api/games/unapproved` Restituisce la lista dei giochi non approvati. (Admin)
  9) `GET /api/games/:id` Restituisce i dettagli di un singolo gioco.
  10) `DELETE /api/games/:id` Elimina un gioco. (Admin)

3.3 Caricamento e Revisione Giochi

Gli utenti registrati possono proporre nuovi giochi compilando un form dedicato. Gli amministratori possono approvare o rifiutare i giochi proposti e caricare direttamente i propri.

3.3.1 Dettagli Implementativi

- **Upload dei file:** Il processo di caricamento di un nuovo gioco inizia da un form, presente in una pagina del frontend dedicata, che l'utente deve compilare con: nome del gioco, descrizione, icona e i file HTML, CSS e JS. Una volta compilato il form, l'utente procede con l'upload tramite l'apposito bottone, che manderà una richiesta all'endpoint di upload (11). Intercettata la richiesta, il backend utilizza la libreria `multer` di Node.js per processare i file e salvarli in una cartella temporanea sul server.

- **Validazione e Salvataggio Atomico**: Una volta ricevuti i dati, il backend esegue una serie di controlli. Se la validazione va a buon fine, i file del gioco vengono spostati dalla cartella temporanea a una cartella dedicata al gioco, viene salvato il gioco nel database e il backend invia una risposta con un messaggio di successo, altrimenti, il backend elimina i file dalla cartella temporanea e invia un messaggio di errore personalizzato.

- **Scaricamento file di gioco:** All'interno della sezione amministrativa, l'admin potrà, tramite apposito bottone, scaricare i file del gioco tramite richiesta all'apposito endpoint (12). Una volta controllato che l'utente loggato sia effettivamente un admin, il backend preparerà, tramite modulo Node.js `archiver`, un file zip della cartella contenente tutti i file del gioco in questione e lo inoltrerà in risposta all'admin, che potrà esaminarne il contenuto.

- **Stato di revisione:** Se al momento del caricamento dei dati del gioco nel database, se l'utente che carica il gioco è admin, il campo `approvedBy` verrà automaticamente popolato dall'id dell'utente autore. In caso contrario, questo verrà lasciato `null`, ovvero in attesa di revisione.
- **Logica di approvazione:** Una volta terminata la propria revisione, l'admin potrà, tramite apposito bottone, approvare il gioco. Questo scatenerà una richiesta all'apposito endpoint (13). Una volta ricevuta la richiesta, il backend, dopo aver controllato se l'utente è admin, modificherà la riga del database relativa al gioco in questione, popolando il campo `approvedBy` con l'id dell'utente amministratore.
- **API Endpoints:**

  11) `POST /api/games/upload` Carica un gioco.

  12) `GET /api/games/download/:id` Scarica i file di un gioco. (Admin)

  13) `PUT /api/games/approve/:id` Approva gioco. (Admin)

3.4 Sistema di Punteggi e Classifiche

Ogni gioco ha una classifica dei 10 migliori punteggi. I punteggi vengono salvati in auotmatico al termine di una partita.

3.4.1 Dettagli Implementativi

- **Comunicazione Gioco-Piattaforma:** Per permettere ai minigiochi di caricare i punteggi in modo sicuro, senza esporre direttamente le API backend, è stata implementata un interfaccia di comunicazione basata sulle web API `postMessage`. Al termine della partita, i minigiochi che vorranno caricare il punteggio della partita dovranno eseguire il seguente codice:

  ```
  window.parent.postMessage({ type: 'submitScore', score: score }, '*');
  ```

  che andrà ad inviare un messaggio alla pagina "genitore" (in questo caso la pagina che contiene il gioco, nel frontend) contenente il tipo di messaggio, `submitScore` e il punteggio, rappresentato come valore numerico. Nel frontend, la pagina del gioco rimarrà in ascolto per ricevere, gestire ed inoltrare i messaggi `submitScore` all'apposito endpoint (14), assieme all'id del gioco e le credenziali utente.
- **Logica di salvataggio:** Una volta ricevuto il punteggio dal frontend, il backend lo valida. Se non rileva problemi, carica il punteggio nel database, assieme all'id dell'utente che lo ha generato, l'id del gioco e la data di caricamento.
- **Recupero della Classifica:** Per ottenere la leaderboard, il frontend esegue una richiesta all'endpoint (15). Per ottimizzare le performance e ridurre il traffico di rete, la query SQL nel backend è progettata per restituire direttamente solo i primi 10 record, ordinati per punteggio in modo decrescente (`ORDER BY score DESC LIMIT 10`), evitando così l'invio di dati superflui al client.
- **API Endpoints:**

  14) `POST /api/games/uploadScore` Carica un punteggio.

  15) `GET /api/games/scores/:id` Ottieni la top 10 dei punteggi di un gioco.

4 Conclusioni

Il progetto LudoTech ha raggiunto con successo gli obiettivi prefissati, realizzando una piattaforma web funzionale per la condivisione e la fruizione di minigiochi. L'architettura sviluppata, basata su un backend Node.js/Express e un frontend Angular/Ionic, si è dimostrata robusta e scalabile per lo scopo e le funzionalità implementate.
Particolare attenzione è stata posta alla sicurezza, la sessione dell'utente, inizialmente gestita tramite Token JWT, è stata modificata del tutto implementando un sistema di autenticazione basato su sessioni tramite cookie HTTP only, una gestione sicura delle password tramite hashing e un meccanismo di revisione per i contenuti generati dagli utenti. La comunicazione isolata tra i giochi e la piattaforma, tramite l'API `postMessage`, rappresenta un punto chiave dell'architettura, garantendo sicurezza e integrità.