StudentHub
Piattaforma per la Gestione della Carriera Universitaria

Diego Andruccioli, Rei Mici, Giovanni Morelli

Anno Accademico 2025/2026

Indice

1 Introduzione

1.1 Contesto e Motivazioni

Nel contesto universitario moderno, la gestione della propria carriera accademica è un aspetto cruciale per ogni studente. Spesso, però, gli strumenti istituzionali si limitano a fornire elenchi statici di voti, senza offrire un feedback immediato sull'andamento globale o proiezioni future. Inoltre, il lungo percorso di studi può portare a cali di motivazione.

StudentHub nasce per rispondere a queste esigenze: è una piattaforma web *Single Page Application* (SPA) progettata per centralizzare la gestione del libretto universitario, trasformando l'inserimento dei dati in un'esperienza gratificante.

1.2 Obiettivi del Progetto

Il progetto si pone due obiettivi principali:

1. **Funzionale**: Fornire allo studente strumenti avanzati di analisi dati (calcolo medie, proiezioni di voto) e incentivare lo studio attraverso meccaniche di *Gamification* (livelli, badge, classifiche).

2. **Tecnico**: Realizzare un sistema robusto basato su un'architettura disaccoppiata (Client-Server), garantendo scalabilità, sicurezza nella gestione dei dati sensibili e un'interfaccia utente moderna e reattiva.

2 Features

In questo capitolo vengono descritte le funzionalità principali della piattaforma. Il sistema implementa un controllo degli accessi basato sui ruoli (RBAC), distinguendo tra funzionalità per studenti e per amministratori.

2.1 Features Utente (Studente)

Gli utenti registrati hanno accesso a una dashboard personale che offre le seguenti funzionalità:

- **Gestione Carriera (CRUD)**: Interfaccia intuitiva per l'inserimento, la modifica e l'eliminazione degli esami. Ogni voce include dettagli su voto, crediti (CFU), data e lode.
- **Data Visualization**: Una dashboard statistica che calcola in tempo reale:
  - Media Aritmetica e Ponderata.
  - Voto di base di laurea proiettato.
  - Grafici interattivi sull'andamento dei voti nel tempo.
- **Sistema di Gamification**: Un motore logico che converte i risultati accademici in progressi di gioco.
  - *Livelli e XP*: Calcolo automatico dei Punti Esperienza basato su un algoritmo che pesa voto e CFU.
  - *Avatar*: Sblocco di nuovi avatar visuali al raggiungimento di specifici livelli.
- **Badge e Obiettivi**: Sistema di *Achievements* automatici (es. "Primo 30 e Lode", "Stacanovista") per gratificare i traguardi raggiunti.
- **Classifica (Leaderboard)**: Sezione sociale che permette il confronto dei propri progressi XP con l'intera base utenti.
- **Personalizzazione e Accessibilità**: Gestione delle preferenze utente, inclusi temi cromatici per l'interfaccia e impostazioni di accessibilità visiva.

2.2 Features Amministrative

Il sistema prevede un'area riservata per la gestione e il monitoraggio della piattaforma, accessibile tramite due livelli di privilegio:

Admin

- **User Management**: Accesso alla lista completa degli utenti iscritti.
- **Ottimizzazione Performance**: La visualizzazione delle liste utenti implementa una *paginazione server-side*, permettendo la gestione fluida anche con migliaia di record.
- **Statistiche Globali**: Visualizzazione di metriche aggregate sull'utilizzo della piattaforma (es. totale esami inseriti, media voti globale).

SuperAdmin

Possiede i massimi privilegi di sistema ed estende le capacità dell'Admin con:

- **Gestione Ruoli**: Possibilità di promuovere utenti standard a ruolo Admin o retrocederli.
- **Gestione Staff**: Possibilità di eliminare account Admin.

3 Tecnologie Utilizzate

3.1 Frontend

Il client è sviluppato utilizzando le seguenti tecnologie:

- **Vue.js** (v3.5.24): Framework progressivo per interfacce utente.
- **Pinia** (v3.0.4): State Management ufficiale e modulare per Vue.js.
- **Vite** (v7.2.4): Build tool di nuova generazione per frontend rapidi.
- **Tailwind CSS** (v4.1.18): Framework CSS utility-first per lo styling.
- **TypeScript (v5.9.3)**: Logica applicativa fortemente tipizzata per garantire sicurezza e scalabilità.

3.2 Backend

Il server è sviluppato utilizzando:

- **Node.js**: Runtime JavaScript lato server.
- **Express.js** (v4.19.2): Web framework per Node.js.
- **TypeScript** (v5.4.5): Superset tipizzato di JavaScript per una maggiore robustezza del codice.

3.3 Database

- **MySQL**: Database relazionale per la persistenza dei dati.

3.4 Containerizzazione

- **Docker & Docker Compose**: Per l'orchestrazione dell'ambiente di sviluppo.

4 Pacchetti Installati

Di seguito vengono elencati i principali pacchetti di terze parti utilizzati.

4.1 Frontend

Installazione dipendenze: `npm install`

`axios` **(^1.13.2)**
: Libreria per effettuare richieste HTTP (API Client) verso il backend in modo centralizzato.

`pinia` **(^3.0.4)**
: Store manager ufficiale per Vue.js, utilizzato per la gestione dello stato globale.

`vue-router` **(^4.6.3)**
: Gestore ufficiale del routing per Vue.js, permette la navigazione tra le pagine (SPA).

`chart.js` **e** `vue-chartjs`
: Librerie per la creazione di grafici statistici interattivi e responsivi.

`tailwindcss`
: Motore per la generazione dei fogli di stile utility-first.

4.2 Backend

Installazione dipendenze: `npm install`

`express` **(^4.19.2)**
: Core framework per la creazione del server web e la gestione delle rotte API.

`mysql2` **(^3.9.7)**
: Driver client ottimizzato per la connessione e l'interazione con il database MySQL.

`bcrypt` **(^5.1.1)**
: Libreria per l'hashing sicuro delle password.

`jsonwebtoken` **(^9.0.2)**
: Strumento per la generazione e validazione dei token JWT.

`cors` **(^2.8.5)**
: Middleware per abilitare e configurare il Cross-Origin Resource Sharing.

`dotenv` **(^16.4.5)**
Modulo per caricare le variabili d'ambiente da un file `.env`.

5 Struttura del Progetto

Di seguito è riportata l'organizzazione delle directory del progetto.

```
StudentHub/
|-- backend/                  # Logica Server
|   |-- src/
|   |   |-- config/           # Configurazioni DB e variabili ambiente
|   |   |-- controllers/      # Gestione richieste HTTP
|   |   |-- middleware/       # Controlli intermedi (es. Autenticazione)
|   |   |-- routes/           # Definizione degli endpoint API
|   |   |-- services/         # Logica di business e accesso ai dati
|   |   |-- types/            # Definizioni dei tipi TypeScript
|   |   |-- utils/            # Funzioni di utilità
|   |-- sql/                  # Script di inizializzazione DB
|   |-- server.ts             # Entry point backend
|
|-- frontend/                 # Interfaccia Utente
|   |-- src/
|   |   |-- api/              # Configurazione Client HTTP (Axios)
|   |   |-- assets/           # Risorse statiche
|   |   |-- components/       # Componenti Vue riutilizzabili
|   |   |-- pages/            # Viste principali
|   |   |-- router/           # Configurazione rotte
|   |   |-- stores/           # Gestione stato (Pinia)
|   |   |-- App.vue           # Componente Root
|   |   |-- main.ts           # Entry point frontend
|
|-- docker-compose.yml        # Configurazione Container
+-- README.md                 # Documentazione progetto
```

6 Installazione

È possibile avviare il progetto tramite Docker (consigliato per compatibilità universale) o manualmente clonando la repository.

Repository GitHub: https://github.com/diegoandruccioli/StudentHub.git

6.1 Metodo 1: Docker (Windows, Mac, Linux)

Requisiti: Docker Desktop installato.

1. Aprire il terminale nella cartella principale del progetto.
2. Eseguire il comando:

   ```
   docker compose up --build
   ```

3. Il sistema sarà accessibile ai seguenti indirizzi:
   - Frontend: http://localhost:5173
   - Backend: http://localhost:3000

6.2 Metodo 2: Installazione Manuale

Prerequisiti: Node.js v18+, MySQL Server in esecuzione.

1. **Configurazione Database**: Eseguire gli script in `backend/sql/` (`init.sql` e `seed.sql`).
2. **Setup Backend**:

   ```
   cd backend
   npm install
   cp .env.example .env # Modificare con credenziali DB
   npm run dev
   ```

3. **Setup Frontend**:

   ```
   cd frontend
   npm install
   npm run dev
   ```

7 API Endpoints

Il backend espone le seguenti API RESTful.

AUTH

- `POST /api/auth/register`: Registra un nuovo utente nel database.
- `POST /api/auth/login`: Effettua il login e restituisce il token di sessione.
- `POST /api/auth/logout`: Effettua il logout invalidando la sessione corrente.

EXAMS

- `GET /api/exams`: Restituisce la lista degli esami dell'utente loggato (con filtri).
- `POST /api/exams`: Aggiunge un nuovo esame alla carriera.
- `PUT /api/exams/:id`: Modifica i dati di un esame esistente.
- `DELETE /api/exams/:id`: Rimuove un esame dalla lista.

STATS

- `GET /api/stats`: Restituisce statistiche aggregate (media, base laurea, proiezioni).

USERS

- `GET /api/users/leaderboard`: Restituisce la classifica globale degli utenti.

GAMIFICATION

- `GET /api/gamification/status`: Livello, XP attuali e progresso.
- `GET /api/gamification/badges`: Catalogo completo degli obiettivi disponibili.
- `GET /api/gamification/my-badges`: Lista degli obiettivi sbloccati dall'utente.

SETTINGS

- `GET /api/settings`: Recupera le preferenze dell'utente (es. CFU totali, tema).
- `PUT /api/settings`: Aggiorna le preferenze dell'utente.

ADMIN

Queste rotte sono protette e accessibili solo a **Admin** o **SuperAdmin**.

- `GET /api/admin/users`: Lista paginata di tutti gli utenti registrati.
- `GET /api/admin/stats/exam-count`: Statistiche globali aggregate sugli esami.
- `GET /api/admin/stats/ranking`: Classifica globale per dashboard admin.
- `PUT /api/admin/users/:id/role`: Modifica del ruolo di un utente (SuperAdmin).
- `DELETE /api/admin/users/:id`: Eliminazione di un account(SuperAdmin).

8 Database

Il sistema utilizza un database relazionale MySQL composto dalle seguenti entità:

- **Utenti**: Memorizza credenziali, dati anagrafici e progressi di gamification (XP).
- **Esami**: Memorizza i singoli esami sostenuti, collegati agli utenti. Include vincoli di integrità per i voti.
- **Livelli**: Tabella di configurazione per le soglie di livello basate sugli XP.
- **Obiettivi**: Tabella di configurazione dei badge ottenibili.
- **Obiettivi_Sbloccati**: Tabella di associazione che traccia i badge ottenuti dagli utenti.
- **Impostazioni_Utente**: Memorizza le preferenze di visualizzazione per ogni utente.

9 Accessibilità

Il progetto pone attenzione all'accessibilità visiva attraverso l'uso di token semantici definiti nel design system. Non vengono utilizzati colori hardcoded, ma variabili CSS (es. `-color-primary`, `-color-accent`) che garantiscono coerenza in tutta l'applicazione. L'utente ha inoltre la possibilità di personalizzare la visualizzazione dei voti tramite le impostazioni, scegliendo tra temi diversi o adattando le soglie cromatiche (es. "Semaforo") alle proprie necessità visive.

10 Mockup

- **Mockup Iniziale (Concept)**: Visualizza su Canva
- **Mockup Finale (Prodotto)**: Visualizza su Canva