Università degli Studi di Salerno

Dipartimento di Informatica

Basi di Dati II

Progetto di Basi di Dati II - ArtLink

Professori
Prof.ssa Genoveffa Tortora
Prof. Luigi Di Biasi

Candidato
Alessandro Carnevale
Matricola: 0522501994

Anno Accademico 2024/2025

Indice

1 Introduzione

Negli ultimi anni, grazie alla digitalizzazione e alla disponibilità di dataset aperti, sono aumentate le opportunità di interazione e fruizione delle opere d'arte. Tuttavia, risulta ancora limitata la possibilità di combinare la consultazione di questi contenuti con strumenti social integrati. Il progetto qui presentato, denominato **ArtLink**, nasce proprio per rispondere a questa esigenza: una piattaforma web basata su tecnologia No-SQL che consente agli utenti di esplorare, condividere e discutere opere d'arte pubblicamente disponibili, valorizzando l'interazione tra utenti e contenuti culturali attraverso funzionalità social avanzate.

1.1 Obiettivi specifici

Il progetto si pone i seguenti obiettivi principali:

- Progettare e implementare una web-app basata su stack **MERN** (MongoDB, Express, React, Node.js) che faciliti la fruizione, la condivisione e l'interazione attorno a opere d'arte digitali di pubblico dominio.
- Integrare il dataset *"The Metropolitan Museum of Art – Open Access"* all'interno della piattaforma, permettendo l'importazione automatica dei metadati essenziali e la presentazione degli asset (immagini) in modo ottimale per il web.
- Favorire la creazione di una community: gli utenti devono poter creare account, mettere "mi piace" alle opere, salvare collezioni personali, aggiungere commenti, cercare artisti/opere per parole chiave e seguire altri membri.

2 Descrizione del Miniworld

Il miniworld di **ArtLink** si articola principalmente intorno alle seguenti entità, organizzate in collezioni No-SQL con MongoDB:

- **User**: utenti registrati con informazioni personali e impostazioni del profilo.
- **Artist**: informazioni sugli artisti, incluse biografie e riferimenti esterni.
- **Artwork**: opere importate automaticamente dall'API del Metropolitan Museum of Art, contenenti metadati dettagliati (titolo, artista, descrizione, immagine).
- **Category**: classificazione delle opere d'arte per facilitare la navigazione tematica.
- **Comment**: commenti inseriti dagli utenti per stimolare la discussione sulle opere.
- **Favorite**: associazione utenti–opere per gestire le preferenze personali.
- **Follow**: relazione tra utenti per gestire dinamiche social come il follow e il feed delle attività.
- **Notification**: notifiche in tempo reale per eventi significativi (like, commenti, follow).

Queste entità sono semanticamente collegate tramite l'uso di campi referenziati (`ObjectId`) e tecniche di popolamento automatico (`populate`). Questo approccio consente di realizzare operazioni di JOIN logiche tra le diverse collezioni.

2.1 Requisiti Funzionali

Gestione Utenti

- **Registrazione e autenticazione:** l'utente può creare un account fornendo e-mail, password e nome visualizzato; autenticazione basata su JWT.
- **Profilo utente:** l'utente autenticato può modificare il proprio profilo (nome, bio, immagine) e visualizzare riepilogo di preferiti e commenti.
- **Visualizzazione profili pubblici:** possibilità di consultare il profilo di altri utenti con lista delle opere da loro commentate o preferite.

Catalogazione e Navigazione delle Opere

- **Visualizzazione catalogo:** utenti guest o autenticati possono sfogliare tutte le opere importate dall'API The Met; per ciascuna vengono mostrati titolo, artista, anno, immagine e categorie.
- **Ricerca e filtri:** ricerca per titolo, artista e tag; filtri per artista, periodo e categoria; ordinamento per data o popolarità.
- **Dettaglio opera:** pagina dedicata con immagine ingrandita, metadati completi, elenco commenti e contatore di preferiti; azioni "Aggiungi ai preferiti" e "Lascia commento".

Interazione Sociale

- **Preferiti:** utenti autenticati possono aggiungere/rimuovere opere dai preferiti; lista visibile nel profilo.
- **Commenti:** inserimento, modifica e cancellazione di commenti con timestamp e autore.
- **Feed attività:** feed personalizzato che mostra ultime attività (commenti, preferiti) degli utenti seguiti.
- **Segui utenti:** funzionalità di follow/unfollow con aggiornamento dinamico del feed.

Notifiche e Real-Time

- **Notifiche:** creazione e invio di notifiche in real time via WebSocket per eventi di like, commenti e follow.

Amministrazione e Moderazione (minimale)

- **Moderazione commenti:** endpoint per rimozione di commenti inappropriati.
- **Gestione categorie:** CRUD di categorie da parte dell'amministratore per aggiornamento tassonomie.

3 Contesto Applicativo

ArtLink si inserisce nel contesto emergente di piattaforme culturali e social dedicate, proponendo una soluzione innovativa e integrata per risolvere il problema della dispersione degli utenti tra repository museali e social network generici. Con **ArtLink**, gli utenti possono esplorare in maniera agevole e interattiva grandi quantità di dati culturali provenienti da istituzioni museali internazionali, favorendo così la nascita e lo sviluppo di community tematiche.

4 Soluzione Proposta

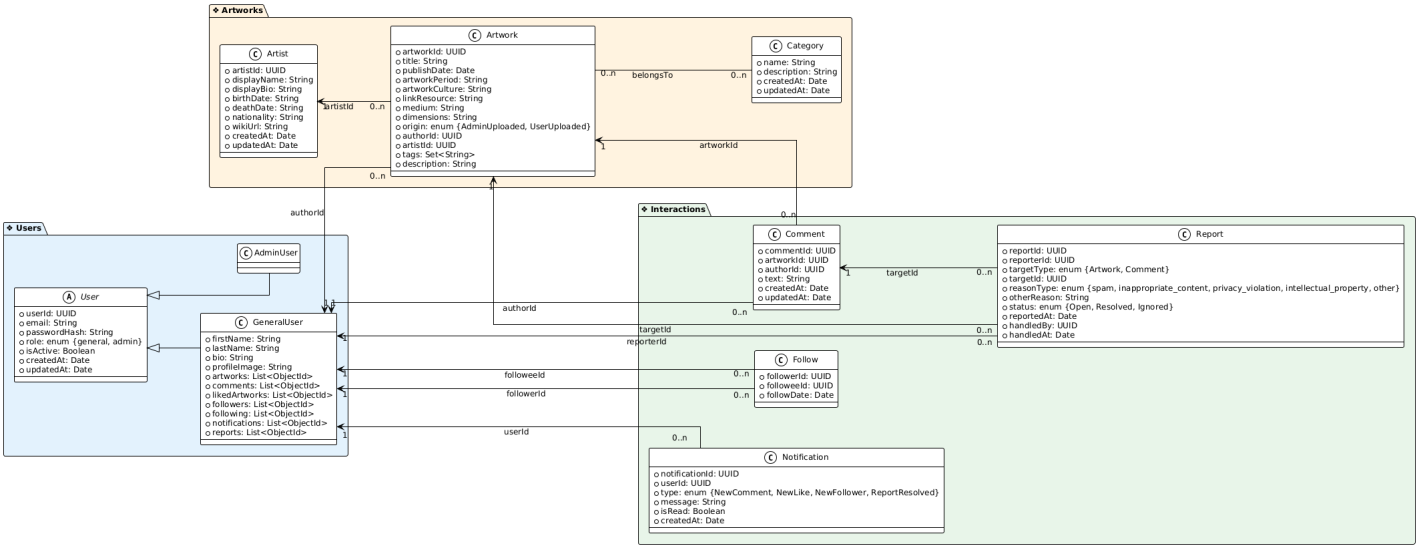

Figura 1: Schema concettuale delle collezioni e delle relazioni nel miniworld di ArtLink: utenti, opere, interazioni e notifiche.

4.1 Architettura Tecnica

La soluzione è basata sullo stack tecnologico *MERN*:

- **MongoDB (No-SQL)**: database con schema flessibile che consente l'ottimizzazione delle query tramite indici e la gestione efficiente delle relazioni tra collezioni.
- **Express e Node.js (Backend)**: implementazione di API RESTful che gestiscono operazioni CRUD, importazione automatizzata di dati tramite API esterne (Met API), autenticazione JWT, e gestione degli eventi di notifica in tempo reale.
- **React (Frontend)**: interfaccia utente responsive e modulare, progettata per garantire un'esperienza utente fluida e intuitiva nella consultazione, interazione e gestione dei contenuti.

4.2 Importazione Dati

Un modulo backend si occupa di importare periodicamente i dati utilizzando l'API del Metropolitan Museum of Art, evitando duplicazioni tramite operazioni di upsert (update o insert).

4.3 Interazioni Utenti e Operazioni CRUD

La piattaforma garantisce operazioni CRUD complete (Create, Read, Update, Delete) per tutte le principali entità (utenti, opere, commenti, preferiti), implementate coerentemente con le proprietà *BASE* (Basically Available, Soft state, Eventual consistency) caratteristiche dei database No-SQL. Queste operazioni sono gestite tramite endpoint protetti con una gestione precisa delle autorizzazioni. Inoltre, il sistema utilizza il meccanismo di `populate` offerto da Mongoose per realizzare operazioni di JOIN logiche tra diverse collezioni, consentendo così il recupero efficiente e combinato delle informazioni correlate. Nella progettazione di queste JOIN, si è scelto di rispettare esplicitamente i vincoli di *Availability* e *Partition tolerance* definiti dal teorema CAP.

4.4 Notifiche in Tempo Reale

Le notifiche vengono gestite tramite documenti dedicati e inviate agli utenti in tempo reale mediante WebSocket, assicurando una comunicazione immediata e continua.

5 Metodologia Utilizzata

La metodologia adottata è articolata nelle seguenti fasi iterative:

- **Analisi dei requisiti**: identificazione delle esigenze principali e delle funzionalità richieste.
- **Progettazione del modello dati**: definizione delle entità e delle relazioni tra collezioni MongoDB, con realizzazione di diagrammi esplicativi.
- **Sviluppo iterativo**:
  - Backend: realizzazione API REST, gestione CRUD, autenticazione e importazione dati.
  - Frontend: creazione interfaccia utente con React, gestione autenticazione e notifiche.
- **Test e Validazione**: verifica delle funzionalità tramite test unitari e test di integrazione con Postman.
- **Documentazione finale**: redazione del presente documento, documentazione tecnica del codice e manuale d'uso per l'installazione.

Questa metodologia ha permesso una gestione efficiente dello sviluppo, con un continuo miglioramento e adattamento delle funzionalità per soddisfare pienamente le esigenze degli utenti finali.