Progetto Sistemi Web - v2

Donaction: Piattaforma Web per la Raccolta Fondi

Introduzione

Donaction è un'applicazione web ideata per facilitare la creazione e la gestione di **campagne di raccolta fondi** per varie cause (ad esempio spese mediche impreviste, emergenze ambientali o aiuti umanitari). L'obiettivo del progetto è di connettere in modo semplice e trasparente chi necessita di supporto economico (**promotori** delle campagne) con coloro che desiderano contribuire alle cause (**donatori**). La piattaforma fornisce un punto d'incontro online dove i promotori possono descrivere i loro progetti e obiettivi di fundraising, mentre i donatori possono effettuare donazioni in modo sicuro e tracciabile.

Donaction è sviluppata come **Single Page Application (SPA)** con front-end in Angular e back-end RESTful in Node.js/Express, seguendo i principi dell'ingegneria del software per le applicazioni web.

Funzionalità Principali

- **Registrazione e Login utenti:** gli utenti possono creare un account fornendo le proprie credenziali (es. email e password) e autenticarsi per accedere alle funzioni dell'app. L'autenticazione garantisce che solo utenti registrati possano avviare campagne o effettuare donazioni.
- **Creazione di campagne di raccolta fondi:** un utente autenticato può assumere il ruolo di promotore e creare una nuova campagna inserendo dettagli come titolo, descrizione, importo obiettivo (*goal*) e categoria/causa. Ogni campagna può includere un'immagine rappresentativa e viene salvata nel sistema associandola al creatore.
- **Visualizzazione e ricerca campagne:** tutti gli utenti (anche non autenticati, in modalità lettura) possono esplorare l'elenco delle campagne attive. È possibile visualizzare la pagina di dettaglio di una singola campagna, che mostra la descrizione estesa, il creatore, l'obiettivo economico e i progressi (quanto è stato raccolto finora). Sono disponibili funzionalità di filtro e ricerca per trovare campagne per parola chiave, categoria o ordinamento (ad esempio, le più recenti o quelle prossime al raggiungimento dell'obiettivo).
- **Donazioni alle campagne:** un utente autenticato con ruolo di donatore può contribuire a una campagna inserendo un importo e confermando la donazione. Il sistema registra la donazione associandola alla campagna selezionata e al profilo dell'utente donatore. *(In un contesto reale, l'app si integrerebbe con un gateway di pagamento; in questo progetto si può simulare l'operazione limitandosi a registrare la transazione.)*
- **Storico delle donazioni:** ogni donatore ha a disposizione una sezione personale dove visualizzare lo storico delle proprie donazioni effettuate, con indicazione della campagna beneficiaria, data e importo. Analogamente, un promotore può consultare la lista delle donazioni ricevute sulle campagne da lui create, per monitorarne l'andamento.
- **Gestione amministrativa:** un utente con ruolo di amministratore ha accesso a funzioni di moderazione e supervisione della piattaforma. Ad esempio, l'admin può visualizzare tutte le campagne e le donazioni effettuate, rimuovere o oscurare campagne inappropriate e gestire i permessi o bloccare utenti che violano le policy.

Ruoli degli Utenti

La piattaforma prevede diversi ruoli utente, ciascuno con permessi specifici. Di seguito sono descritti i tre ruoli principali e le relative responsabilità e funzionalità consentite:

Ruolo	Descrizione	Azioni principali
Donatore	Utente registrato che effettua donazioni alle campagne di raccolta fondi.	• Visualizzare le campagne disponibili • Effettuare donazioni alle campagne • Consultare il proprio storico di donazioni
Promotore	Utente (registrato) che crea e promuove campagne di raccolta fondi (spesso coincidente con il beneficiario dei fondi).	• Creare nuove campagne specificandone l'obiettivo • Modificare/aggiornare le campagne create (es. modificarne la descrizione o chiudere la campagna una volta raggiunto l'obiettivo) • Visualizzare le donazioni ricevute sulle proprie campagne
Amministratore	Utente con privilegi elevati incaricato di gestire la piattaforma.	• Visualizzare e monitorare **tutte** le campagne e donazioni • Moderare contenuti inappropriati (es. rimuovere campagne non conformi) • Gestire gli account utente (es. bloccare utenti malevoli)

Architettura del Sistema

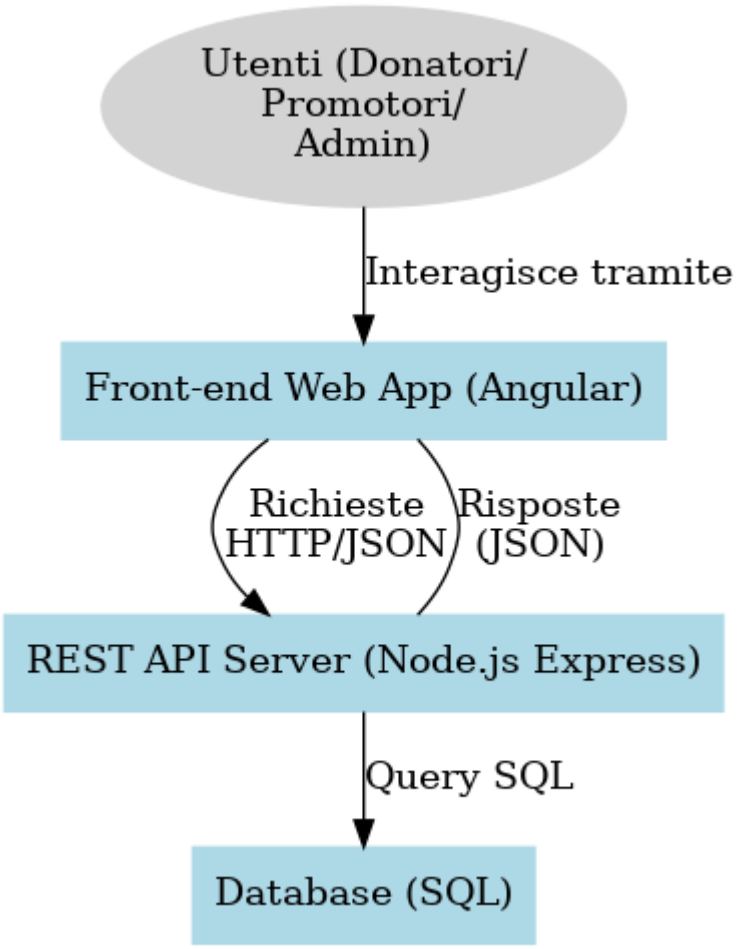

Figura: Architettura a tre livelli di Donaction. L'applicazione segue una classica **architettura client-server a tre livelli**, separando nettamente interfaccia utente, logica applicativa e gestione dei dati. Il **front-end** è una Single Page Application Angular che gira nel browser dell'utente e fornisce l'interfaccia interattiva (moduli di iscrizione, pagine di elenco e dettaglio campagne, form per donare, ecc.). Il **back-end** è un server Node.js con framework Express che espone una **API RESTful**: esso riceve le richieste HTTP dal front-end (ad esempio richieste per ottenere l'elenco delle campagne o effettuare una donazione), le elabora applicando la logica di business e interagisce con il database, quindi restituisce al client i dati richiesti in formato JSON. Infine, il **database relazionale** (ad esempio MySQL) rappresenta il livello di **storage** persistente, conservando le informazioni su utenti, campagne e donazioni.

Questa architettura a livelli segue i principi di **separazione delle responsabilità**: la parte client è dedicata alla presentazione e all'esperienza utente, la parte server gestisce la logica, l'autenticazione e il coordinamento dei dati, mentre il database si occupa di memorizzazione e consistenza dei dati. La comunicazione avviene tramite chiamate HTTP *stateless* al server (API REST) – ciò rende il sistema più scalabile e manutenibile, permettendo ad esempio di sostituire o aggiornare un componente senza impattare sugli altri. Inoltre, un'API REST standardizzata potrebbe consentire a diversi tipi di client (come applicazioni mobile) di interagire con il sistema utilizzando le stesse interfacce di servizio.

Modello dei Dati

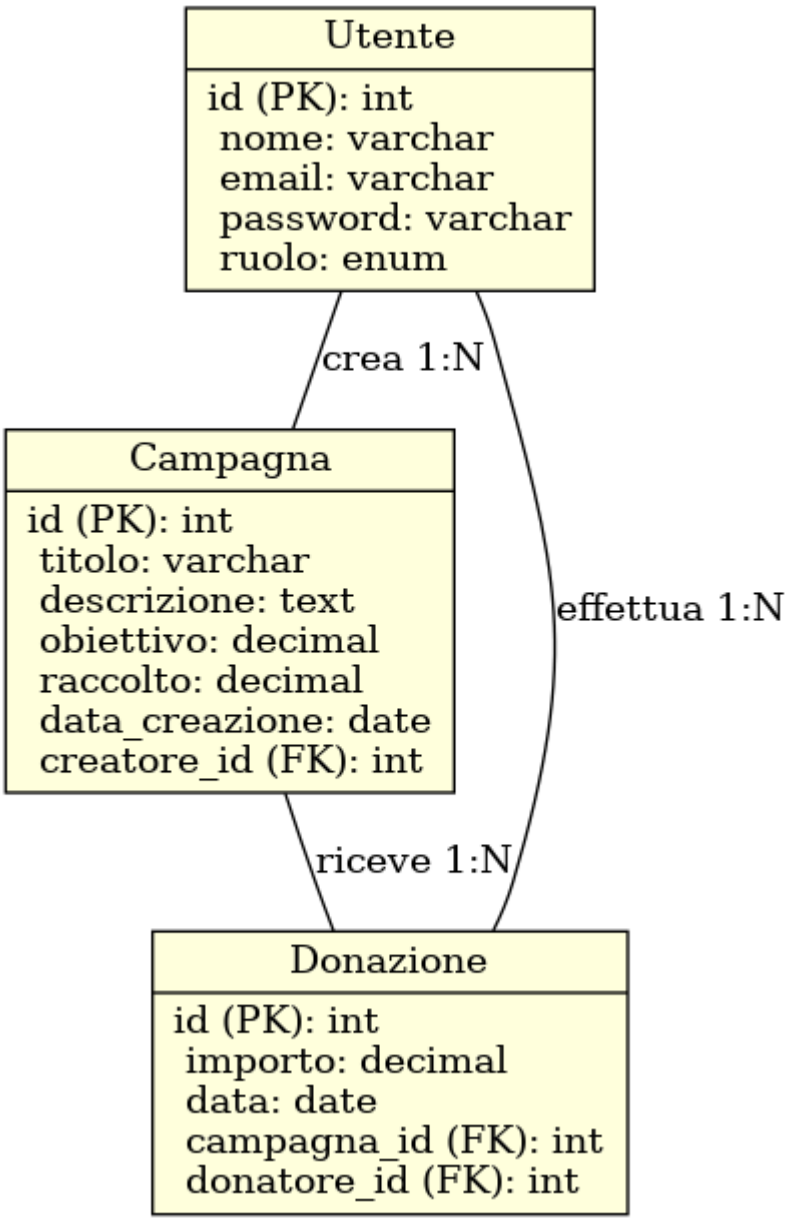

Figura: Diagramma ER (Entità-Relazioni) del modello dati di Donaction. Il modello dati è progettato attorno a tre entità principali:

- **Utente:** rappresenta un utilizzatore registrato della piattaforma. Gli attributi fondamentali includono un ID univoco, nome, email (univoca per login), password (salvata in forma cifrata) e ruolo (ad esempio *donatore*, *promotore* o *admin*).
- **Campagna:** rappresenta una campagna di raccolta fondi creata da un utente promotore. Contiene campi come ID univoco, titolo, descrizione, importo obiettivo da raccogliere, importo attualmente raccolto, data di creazione e un riferimento al **creatore** (l'ID dell'utente promotore che l'ha lanciata). Facoltativamente può includere una categoria/causa per facilitare la ricerca (es. *sanità*, *ambiente*, *educazione*).
- **Donazione:** rappresenta una singola donazione effettuata da un utente a favore di una campagna. Gli attributi includono ID univoco, importo donato, data/ora della donazione, riferimento al **donatore** (ID utente che ha donato) e riferimento alla **campagna** beneficiaria.

Le relazioni tra le entità sono indicate nel diagramma ER: un singolo utente può creare più campagne (relazione 1:N tra Utente e Campagna) ed effettuare più donazioni (relazione 1:N tra Utente e Donazione). Ogni campagna è creata da un utente (N:1 verso Utente) e può ricevere molte donazioni da donatori diversi (1:N verso Donazione). Analogamente, ogni donazione appartiene esattamente a una campagna (N:1 verso Campagna) ed è effettuata da un singolo utente donatore (N:1 verso Utente). In questo modo si realizza indirettamente una relazione molti-a-molti tra Utenti e Campagne attraverso l'entità Donazione (un utente può donare a più campagne e una campagna può ricevere donazioni da più utenti).

A livello di **schema relazionale**, le tre entità corrispondono a tre tabelle (Utente, Campagna, Donazione) collegate da **chiavi esterne**. Ad esempio, la tabella *Campagna* contiene un attributo `creatore_id` che è chiave esterna verso la tabella *Utente* (collegato all'ID utente), mentre la tabella *Donazione* registra `campagna_id` (FK verso Campagna) e `donatore_id` (FK verso Utente). Queste relazioni assicurano l'integrità referenziale (non si può avere una donazione senza un utente valido e una campagna esistente). Indici e chiavi primarie sono definiti su tutti gli ID per ottimizzare le operazioni di ricerca e join sui dati del sistema.

Principi di Ingegneria del Software Applicati

- **Separazione dei livelli e modularità:** l'applicazione è sviluppata seguendo un'architettura multilivello chiara (front-end, back-end, database) e un design modulare. Ciò migliora la **manutenibilità** e la **riusabilità** del codice: ad esempio, nel front-end Angular le funzionalità sono suddivise in componenti e servizi riutilizzabili, mentre nel back-end Express il codice è organizzato in router, controller e moduli service per separare la logica di business, la gestione delle API e l'accesso ai dati.
- **Gestione della sessione (autenticazione):** per mantenere le sessioni utente in modo **scalabile**, Donaction adotta un meccanismo **stateless** basato su token JWT (JSON Web Token). Dopo il login, al client viene fornito un token di autenticazione che viene incluso in ogni richiesta verso l'API. Il server valida il token (che contiene l'ID utente e il ruolo) ad ogni richiesta protetta, evitando di conservare stato di sessione lato server. Questo approccio semplifica il design e supporta la scalabilità, poiché *le applicazioni stateless non conservano le informazioni di sessione dell'utente tra le interazioni*, trattando ogni richiesta in modo indipendente.
- **Scalabilità e prestazioni:** grazie alla natura stateless del server REST e all'uso di tecnologie asincrone (Node.js), l'applicazione può scalare orizzontalmente con facilità. È possibile eseguire istanze multiple del server API dietro un load balancer per gestire un alto numero di richieste contemporanee, senza necessità di condivisione di sessione tra server. Anche il database può essere ottimizzato mediante indici e, se necessario, *replicato* per bilanciare il carico in lettura. L'uso di Angular sul client riduce il carico sul server, rendendo l'interfaccia reattiva e spostando parte dell'elaborazione (come la validazione dei form) sul lato utente.
- **Sicurezza e robustezza:** il progetto adotta diverse misure di sicurezza a livello applicativo. Le password utente vengono memorizzate solo in forma **hash** (ad esempio con algoritmo Bcrypt) anziché in chiaro, proteggendo le credenziali in caso di violazione di dati. Le API implementano controlli di **autorizzazione** basati sul ruolo utente (ad esempio solo gli admin possono cancellare o modificare una campagna altrui). Si effettuano validazioni sia lato client (nei form Angular) sia lato server (nei controller Express) sugli input, per prevenire inconsistenze o attacchi come SQL injection. Inoltre, la comunicazione tra front-end e server avviene su protocollo **HTTPS**, garantendo la cifratura dei dati sensibili in transito.

Scelte Tecnologiche

- **Front-end: Angular** – Sviluppato il front-end con Angular (framework SPA in TypeScript, es. versione 15) per sfruttare binding bidirezionale, routing client-side e un ricco ecosistema di componenti UI e servizi. Angular accelera lo sviluppo di interfacce **responsive** e modulari, facilitando la manutenzione del codice client.
- **Back-end: Node.js & Express** – L'API REST è costruita con Node.js (runtime JavaScript lato server) e il framework web Express. Questa scelta consente di utilizzare lo stesso linguaggio (TypeScript/JavaScript) su front-end e back-end, semplificando lo sviluppo. Express è leggero e adatto a creare rapidamente endpoint RESTful, mentre Node.js (architettura **event-driven**, non bloccante) gestisce con efficienza molte richieste concorrenti, caratteristica importante per un'app di crowdfunding con potenzialmente numerosi accessi contemporanei.
- **Database: MySQL** – Per la persistenza dei dati si è optato per un database relazionale SQL (ad esempio MySQL o PostgreSQL). Un database relazionale è adatto a rappresentare le relazioni tra utenti, campagne e donazioni garantendo integrità tramite vincoli (chiavi primarie/esterne) e supportando interrogazioni efficienti in SQL (ad es. ottenere lo storico donazioni di un utente tramite join su tabelle). MySQL, in particolare, è ampiamente collaudato, supporta transazioni (utile ad esempio per assicurare che l'inserimento di una donazione e l'aggiornamento dell'importo raccolto avvengano in modo atomico) ed è facilmente integrabile con Node.js (tramite driver o ORM come Sequelize).
- **API RESTful & JSON:** la comunicazione tra client Angular e server avviene tramite API REST, con dati formattati in JSON. Questo standard aperto rende il sistema interoperabile e facile da debuggare. Le richieste HTTP seguono convenzioni REST (GET per leggere dati, POST per creare, PUT/PATCH per aggiornare, DELETE per eliminare) e le risposte includono codici di stato appropriati (es. 200 OK, 201 Created, 401 Unauthorized), facilitando l'interazione standardizzata.
- **Strumenti e librerie:** il progetto utilizza **npm** (Node Package Manager) per la gestione delle dipendenze sia lato client che server. Alcuni pacchetti chiave includono *Cors* (abilitare chiamate API cross-domain), *jsonwebtoken* (gestione JWT per autenticazione), *Bcrypt* (hashing delle password) e librerie di interfaccia come *Angular Material* per componenti grafici predefiniti. L'ambiente Angular CLI integra *Webpack* per il bundling del codice front-end, mentre in sviluppo lato server si impiega *Nodemon* per ricaricare automaticamente l'applicazione ad ogni modifica del codice.

Possibili Estensioni Future

- Integrazione di **gateway di pagamento** reali: implementare connessioni a servizi come PayPal, Stripe o Satispay per processare effettivamente i pagamenti con carta di credito o altri metodi. Ciò permetterebbe che le donazioni registrate corrispondano a transazioni reali (gestendo conferme di pagamento, ricevute, ecc.).
- **Notifiche e comunicazioni:** aggiungere un sistema di notifica (email o push) per informare i promotori quando arriva una nuova donazione, e per aggiornare i donatori sugli sviluppi delle campagne a cui hanno contribuito (ad esempio, notifica al raggiungimento dell'obiettivo della campagna).
- **Funzionalità social e condivisione:** permettere la condivisione diretta delle campagne sui social network, per aumentarne la visibilità. Si potrebbero anche implementare commenti o sezioni di aggiornamento in cui il promotore comunica il progresso della raccolta e ringrazia i sostenitori.
- **Ottimizzazioni UI/UX e mobile:** sviluppare una versione mobile dedicata o una Progressive Web App, per facilitare l'accesso da smartphone e abilitare notifiche push. Inoltre, migliorare l'esperienza utente con suggerimenti personalizzati di campagne per i donatori e filtri più avanzati (ad esempio per area geografica o tipologia di causa).

Conclusioni

In **conclusione**, Donaction offre una soluzione mirata e ben progettata per gestire campagne di crowdfunding in maniera efficace. La combinazione di funzionalità essenziali, architettura scalabile e scelte tecnologiche moderne garantisce che la piattaforma soddisfi i requisiti del progetto d’esame, dimostrando concretamente l’applicazione dei principi di ingegneria dei sistemi web. Il risultato è un’app web completa ma realizzabile, che può costituire la base per ulteriori sviluppi e miglioramenti futuri.