Dipartimento di
Ingegneria Gestionale, dell'Informazione e della Produzione

Corso di Laurea in
Ingegneria Informatica

Progettazione algoritmi e computabilità

Giorgio Passarella
Matricola n. 1079287

Nome Secondo
Matricola n. XXXXXX

Nome Terzo
Matricola n. YYYYYY

Anno Accademico
2025/2026

Indice

Elenco delle figure

Elenco delle tabelle

Listings

Capitolo 1

Iterazione 0

1.1 Introduzione

BuddyMaps è un sistema intelligente di ottimizzazione dei viaggi progettato per risolvere il problema logistico che ogni turista affronta: scegliere cosa visitare è facile, ma decidere come farlo nel minor tempo possibile è una sfida complessa.

A differenza delle tradizionali applicazioni di mappatura, BuddyMaps non si limita a mostrare la posizione dei luoghi, ma trasforma una lista disordinata di punti di interesse in un itinerario perfetto. L'applicazione permette all'utente di selezionare dai 5 ai 10 monumenti e calcola automaticamente il percorso più breve per visitarli tutti e tornare al punto di partenza.

Con BuddyMaps, l'utente può:

- **Pianificare Itinerari Smart:** Selezionare i punti di interesse desiderati e lasciare che il sistema calcoli il ciclo ottimale, riducendo drasticamente i chilometri inutili e i tempi di percorrenza.
- **Gestire l'Itinerario in modo flessibile:** Grazie al modulo di *Gestione Itinerario*, è possibile creare nuovi viaggi, modificarne le tappe, visualizzare i dettagli o eliminare percorsi non più necessari.
- **Consultare lo Storico:** Salvare i percorsi preferiti o quelli già effettuati per poterli riutilizzare o consultare in futuro, mantenendo traccia delle proprie esperienze di viaggio.

L'applicazione si interfaccia direttamente con Provider di Servizi Geografici esterni per garantire che il calcolo delle distanze e dei tempi sia basato su dati reali e aggiornati.

Perché scegliere BuddyMaps?

BuddyMaps nasce per rispondere a un'esigenza specifica: l'efficienza. Spesso i turisti perdono tempo prezioso camminando avanti e indietro per la città a causa di una pianificazione non ottimale. Questo problema, noto in informatica come *Traveling Salesman Problem* (TSP), diventa difficile da risolvere mentalmente man mano che i luoghi da visitare aumentano.

BuddyMaps automatizza questo processo complesso utilizzando algoritmi avanzati sui Grafi e euristiche dedicate. Il sistema non offre solo un percorso, ma il *miglior* percorso possibile, rendendolo lo strumento ideale per diverse tipologie di utenti: dal "Turista Efficiente" che vuole massimizzare le visite in un solo giorno, al "Runner Urbano" che cerca un percorso di allenamento circolare che tocchi specifici landmark, fino ai professionisti dell'ospitalità che desiderano offrire itinerari personalizzati ai propri ospiti.

Con un'interfaccia intuitiva che separa nettamente la logica di presentazione da quella algoritmica, BuddyMaps offre una soluzione potente ma accessibile per trasformare il modo in cui esploriamo le città.

1.2 Requisiti

Il sistema deve soddisfare i seguenti requisiti funzionali e non funzionali per garantire un'esperienza di pianificazione ottimale, come delineato nel diagramma dei casi d'uso:

Requisiti Funzionali

- **Gestione Autenticazione e Profilo:** Gli utenti devono poter effettuare la registrazione, il login e il logout. Il sistema deve permettere la gestione del profilo includendo la modifica dei dati personali, il cambio password e l'eliminazione definitiva dell'account.
- **Visualizzazione e Catalogo:** Il sistema deve offrire una vista astratta per la visualizzazione dei percorsi, permettendo all'utente di sfogliare un catalogo, applicare filtri di ricerca ed esportare l'itinerario desiderato.
- **Creazione Itinerario Manuale:** L'utente può generare un nuovo viaggio impostando un punto di partenza e selezionando manualmente i monumenti e le tappe d'interesse.
- **Creazione Itinerario Casuale:** Generazione automatica basata su percorsi esistenti o nuovi, con la possibilità di definire tempistiche specifiche per la visita.
- **Ottimizzazione del Percorso (TSP):** Il sistema deve integrare un modulo di ottimizzazione che, interfacciandosi con un Provider di Servizi Geografici esterno, calcoli il percorso ottimo. Tale ottimizzazione è inclusa automaticamente durante la creazione o la modifica delle tappe.
- **Gestione e Stato Viaggi:** Gli utenti devono poter salvare percorsi, modificarne le tappe, eliminarli o contrassegnarli come "completati".
- **Social e Storico:** Il sistema deve gestire uno storico dei viaggi che permetta la condivisione degli itinerari e l'aggiunta di recensioni per i percorsi effettuati.

Requisiti Non Funzionali

- **Architettura Modulare:** Il sistema deve separare nettamente le funzionalità di interfaccia (Vista User) dalle logiche di elaborazione e integrazione esterna (Vista Sistema).
- **Usabilità:** L'interfaccia deve permettere l'estensione fluida di funzionalità secondarie (es. filtri, esportazione) senza interrompere il flusso principale di creazione del viaggio.

- **Efficienza e Accuratezza:** Il calcolo dell'ottimizzazione deve essere tempestivo e basato su dati geografici reali forniti da terze parti.
- **Sicurezza:** Garantire l'accesso protetto alle funzionalità di gestione profilo e la persistenza sicura dei percorsi salvati e dello storico.

1.3 Casi D'uso

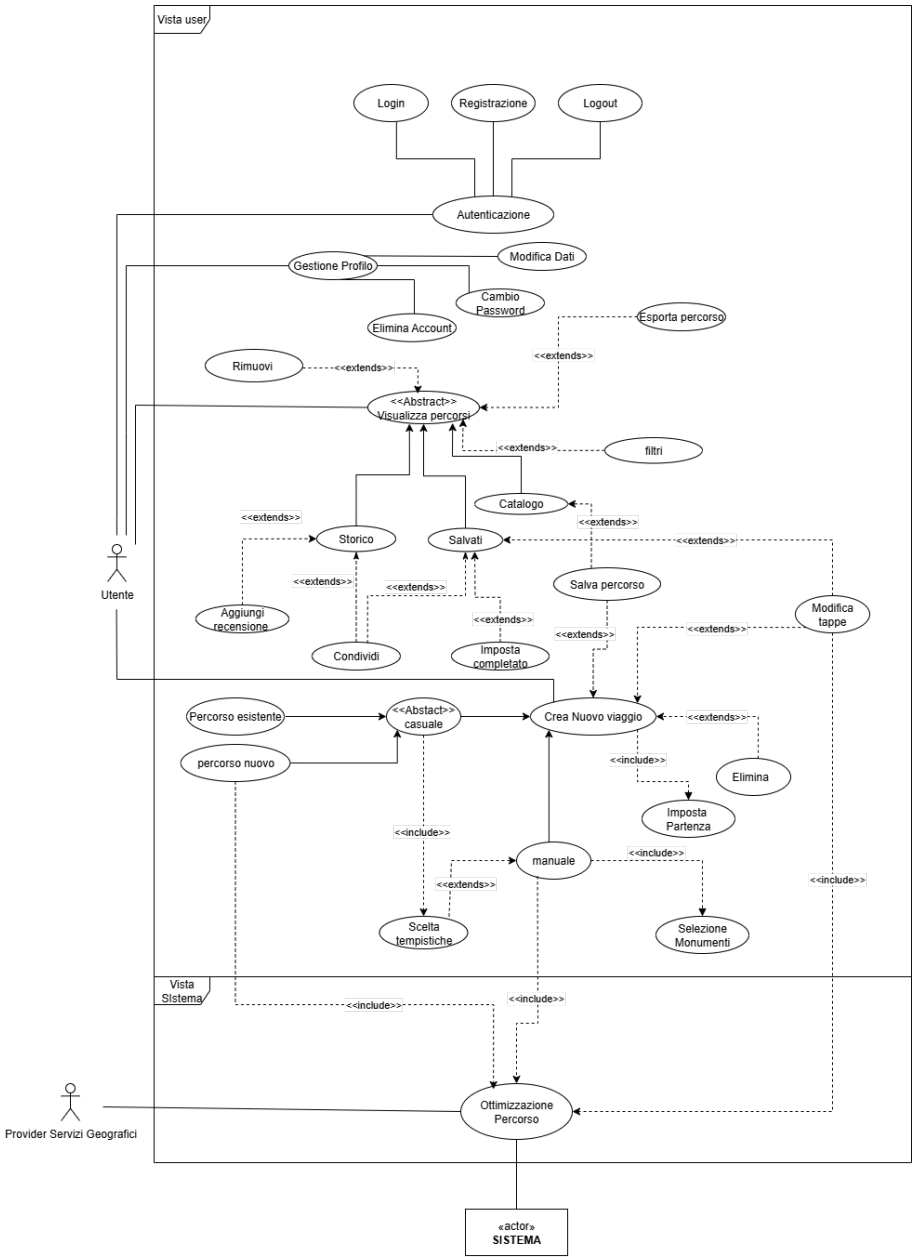

Figura 1.1: Casi d'uso

Di seguito sono riportati i casi d'uso relativi al sistema di gestione percorsi e viaggi.

- **UC1 - Registrazione:** Come nuovo utente, voglio potermi registrare al sistema per creare un account personale.
- **UC2 - Login:** Come utente registrato, voglio poter accedere al mio account per utilizzare le funzionalità del sistema.
- **UC3 - Logout:** Voglio poter terminare la mia sessione in modo sicuro.

- **UC4 - Autenticazione:** Voglio che il sistema verifichi le mie credenziali per consentire l'accesso sicuro.
- **UC5 - Gestione profilo:** Voglio poter gestire le informazioni del mio profilo personale.
- **UC6 - Modifica dati:** Voglio poter modificare i miei dati personali.
- **UC7 - Cambio password:** Voglio poter aggiornare la mia password per motivi di sicurezza.
- **UC8 - Elimina account:** Voglio poter eliminare definitivamente il mio account.
- **UC9 - Visualizza percorso:** Voglio poter visualizzare i dettagli di un percorso.
- **UC10 - Catalogo percorsi:** Voglio poter consultare un catalogo di percorsi disponibili.
- **UC11 - Filtri ricerca:** Voglio poter filtrare i percorsi secondo determinati criteri.
- **UC12 - Salva percorso:** Voglio poter salvare un percorso tra i miei preferiti.
- **UC13 - Visualizza percorsi salvati:** Voglio poter visualizzare l'elenco dei percorsi salvati.
- **UC14 - Visualizza storico:** Voglio poter visualizzare lo storico dei percorsi effettuati.
- **UC15 - Esporta percorso:** Voglio poter esportare un percorso in un formato condivisibile.
- **UC16 - Rimuovi percorso salvato:** Voglio poter rimuovere un percorso dai miei salvati.
- **UC17 - Condividi percorso:** Voglio poter condividere un percorso con altri utenti.
- **UC18 - Aggiungi recensione:** Voglio poter aggiungere una recensione a un percorso.
- **UC19 - Imposta percorso completato:** Voglio poter segnare un percorso come completato.
- **UC20 - Crea nuovo viaggio:** Voglio poter creare un nuovo viaggio personalizzato.
- **UC21 - Scegli percorso esistente:** Voglio poter creare un viaggio partendo da un percorso già esistente.

- **UC22 - Crea percorso nuovo:** Voglio poter creare un percorso completamente nuovo.
- **UC23 - Genera percorso casuale:** Voglio poter generare automaticamente un percorso casuale.
- **UC24 - Imposta punto di partenza:** Voglio poter scegliere il punto di partenza del viaggio.
- **UC25 - Selezione monumenti:** Voglio poter selezionare i monumenti o punti di interesse da includere nel percorso.
- **UC26 - Inserimento manuale tappe:** Voglio poter inserire manualmente le tappe del percorso.
- **UC27 - Scelta tempistiche:** Voglio poter definire le tempistiche del viaggio.
- **UC28 - Ottimizzazione percorso:** Voglio che il sistema ottimizzi automaticamente l'ordine delle tappe.
- **UC29 - Modifica tappe:** Voglio poter modificare le tappe di un percorso creato.
- **UC30 - Elimina tappa:** Voglio poter eliminare una tappa dal percorso.

1.4 Deployment Diagram

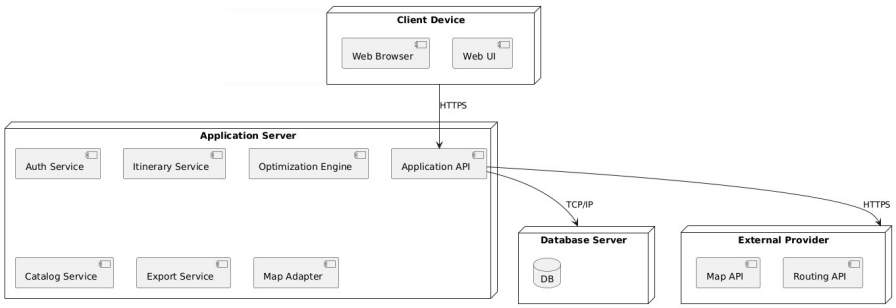

Figura 1.2: Diagramma di deployment

La Figura mostra l'organizzazione del sistema secondo un modello multilivello che separa l'interfaccia utente, i servizi applicativi e le integrazioni esterne.

- **Livello Client** L'accesso al sistema avviene tramite un browser web. La Web UI comunica con il backend attraverso richieste `HTTPS` indirizzate all'Application API. Questo livello si occupa esclusivamente della presentazione e dell'interazione con l'utente finale.

- **Application Server** Costituisce il cuore dell'applicazione e ospita diversi servizi interni:

  - **Auth Service**: gestisce autenticazione e autorizzazione;
  - **Itinerary Service**: gestisce la creazione e modifica degli itinerari;
  - **Optimization Engine**: calcola il percorso ottimale tra i monumenti selezionati;
  - **Catalog Service**: fornisce i dati relativi ai monumenti e ai punti di interesse;
  - **Export Service**: genera output esportabili;
  - **Map Adapter**: gestisce le comunicazioni con i servizi di mappe esterni.

  Tutti i servizi comunicano tramite l'Application API, che funge da punto di orchestrazione.

- **Database Server** Ospita il database principale dell'applicazione e comunica con l'Application API. Gestisce dati persistenti come utenti, monumenti, itinerari e preferenze.

- **External Provider** Il sistema si integra con servizi esterni per ottenere informazioni geografiche e di navigazione:

  - **Map API**: fornisce mappe, geocodifica e dati geografici;
  - **Routing API**: calcola distanze e tempi di percorrenza, supportando l'algoritmo di ottimizzazione.

1.5 Architettura del Sistema

Il sistema segue il pattern architetturale **Model-View-Controller (MVC)**, una scelta progettuale che permette di separare nettamente la logica di presentazione, la logica di ottimizzazione degli itinerari e la gestione dei dati. Tale separazione favorisce la modularità del codice, facilitando la manutenzione e l'estendibilità del software nel tempo.

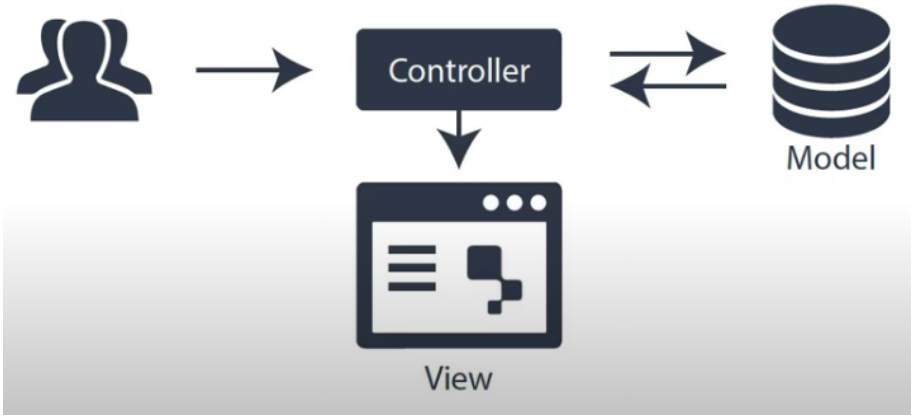

Figura 1.3: Archiettura MVC del Sistema

Model costituisce il nucleo logico e strutturale del sistema, responsabile della gestione dei dati, delle regole di dominio e dei meccanismi di persistenza dei dati. In questo sistema:

- **Database NoSQL:** I dati sono memorizzati in MongoDB. La scelta di un database non relazionale è dettata dalla necessità di gestire con flessibilità i Points of Interest (POI) e i dati geografici. Per la gestione e la visualizzazione dei dati a livello amministrativo, viene utilizzata la GUI MongoDB Compass
- **Backend in Java:** Il core del sistema è implementato utilizzando il framework Spring Boot, che orchestra i servizi attraverso un'architettura orientata ai componenti.
- **Integrazione Geografica:**
  - **OpenStreetMap:** Utilizzato come provider per le coordinate geografiche e i metadati dei POI.
  - **GraphHopper:** Integrato come motore di routing per il calcolo del percorso ottimale tra i punti selezionati.
- **Data Access Layer:** L'interazione con il database avviene tramite Spring Data MongoDB.

View è responsabile dell'interfaccia utente e della visualizzazione dei dati elaborati dal sistema:

- **React.js:**Il frontend è sviluppato con la libreria React, utilizzando componenti funzionali e JSX per una UI reattiva. Lo sviluppo avviene tramite Visual Studio Code, ottimizzato per la gestione dei pacchetti e il debug del frontend

- **Mapping e Visualizzazione:** Per la resa grafica delle mappe e dei percorsi viene utilizzata react-leaflet, permettendo di visualizzare i layer di OpenStreetMap direttamente nel browser.
- **Ambiente di Esecuzione:** L'applicazione frontend viene gestita tramite Node.js, utilizzato per la gestione delle dipendenze e l'avvio del server di sviluppo locale.
- **Gestione API:** Le chiamate asincrone verso il backend sono affidate ad Axios, garantendo uno scambio dati fluido tra client e server.

controller funge da intermediario tra il Model e la View, gestendo le richieste dell'utente e ottimizzando la logica di interazione:

- **API:** Il backend espone endpoint documentati tramite i Controller di Spring Boot. Ogni richiesta HTTP inviata dal frontend tramite Axios viene gestita dal metodo specifico.
- **ottimizzazione:** Il controller riceve in input coordinate di partenza e di arrivo, interroga i servizi di routing e restituisce i risultati elaborati, in base al percorso ottimizzato.
- **Formato Dati:** Lo scambio di informazioni tra client e server avviene esclusivamente in formato JSON.