Relazione progetto Ingegneria dei Sistemi Web

Francesco Santi

Umberto Schiavone

PetPlanet

Introduzione: Il progetto propone lo sviluppo di un'applicazione Single Page Application (SPA) utilizzando tecnologie come Angular, TypeScript, HTML, CSS per il frontend, e Fastify, Node.js, JavaScript come stack di sviluppo per il backend. MongoDB è stato selezionato come database e implementato all'interno di un container Docker insieme a Mongoose. L'architettura adottata segue un approccio di separazione delle responsabilità, mirando a garantire una struttura chiara e modulare per l'applicazione.

Architettura del Sistema:

L'architettura del sistema si basa sul principio fondamentale di separazione delle responsabilità, che sottolinea la suddivisione del sistema in componenti distinti. Angular assume il ruolo di gestire il frontend, mentre Fastify si occupa del backend. La comunicazione tra le due componenti avviene attraverso API ben definite, consolidando un approccio che agevola la gestione efficiente delle operazioni e mantiene chiare le responsabilità di ciascun componente.

Nel backend, la logica di implementazione è stata strutturata mediante l'utilizzo di diverse classi, che comprendono:

1. **Model:**
    - In questa classe, sono gestite le diverse entità del sistema, tra cui utente, amici, animali, post e followers. Le relazioni tra queste entità sono stabilite attraverso la classe utente, utilizzando una proprietà di Mongoose.
2. **Routes:**
    - La classe routes è responsabile della gestione delle diverse rotte all'interno del backend. Queste rotte consentono l'invocazione delle API e l'utilizzo dei metodi associati a ciascuna entità.
3. **Controllers:**

- La classe controllers è stata aggiunta per implementare i vari metodi di gestione delle API, includendo le operazioni CRUD (Create, Read, Update, Delete). Questi metodi consentono di manipolare le entità del sistema in modo coerente e strutturato.

L'implementazione di questo approccio basato su classi nel backend promuove una struttura organizzata e modulare. La chiara distinzione tra model, routes e controllers facilita la manutenzione del codice, permettendo una gestione più agevole e una comprensione immediata delle funzionalità svolte da ciascuna classe.

In sintesi, l'architettura del sistema riflette non solo la separazione delle responsabilità tra frontend e backend ma incorpora anche una struttura ben definita nel backend, basata su classi, per gestire in modo efficiente le entità del sistema attraverso le API.

Sviluppo Agile:

Durante la fase iniziale del progetto, ci siamo dedicati a una fase di brainstorming per definire il concetto di base dell'applicazione. Abbiamo valutato diverse idee, considerando aspetti come l'usabilità, l'interattività e l'attrattiva per gli utenti. Dopo un'attenta analisi, abbiamo optato per la realizzazione di un'applicazione Social basata su una Single Page Application.

La selezione del linguaggio di programmazione e dei framework è stata un punto cruciale nella definizione dell'architettura del progetto. Abbiamo fatto la scelta di Angular per il frontend e di Fastify insieme a Node.js per il backend. Questa decisione strategica è stata guidata dalla volontà di sfruttare le peculiarità di TypeScript per il frontend, garantendo al contempo coerenza e semplificazione nella manutenzione del codice.

In particolare, TypeScript è stato adottato esclusivamente nel frontend per sfruttare i suoi vantaggi nella gestione dei tipi e nell'offrire una struttura più robusta al codice. Per il backend, abbiamo utilizzato JavaScript, mantenendo un equilibrio tra efficacia nello sviluppo e adattamento alle esigenze specifiche della parte server dell'applicazione.

L'utilizzo di Angular per il frontend e di Fastify con Node.js per il backend ha consentito una sinergia efficace tra le due componenti dell'applicazione. Angular ha agevolato lo sviluppo di un'interfaccia utente intuitiva e dinamica, mentre Fastify e Node.js hanno fornito una solida base per la gestione efficiente del backend.

Abbiamo deciso di adottare un approccio Agile nello sviluppo del progetto, ponendo particolare enfasi sull'implementazione delle user stories fin dalle prime fasi del ciclo di

sviluppo. Questo approccio ci ha consentito di rispondere in modo flessibile ai cambiamenti nei requisiti e di mantenere un dialogo continuo per garantire il successo del progetto.

La fase di progettazione ha visto la creazione di mockup utilizzando Balsamiq, un'importante tappa che ci ha permesso di visualizzare l'aspetto e il flusso dell'applicazione prima di passare allo sviluppo effettivo.

Per gestire il processo di sviluppo e coordinare le attività del team, abbiamo adottato GitHub come sistema di controllo versione. La creazione di una pagina dedicata ai task ci ha permesso di tenere traccia delle attività in corso, garantendo una visione chiara dello stato del progetto. Inoltre, abbiamo sfruttato la funzionalità di creazione di branch e merge di GitHub per gestire le diverse fasi di sviluppo in modo efficiente.

La suddivisione del lavoro attraverso la creazione di branch specifici ha facilitato la collaborazione all'interno del team, consentendo a ciascun membro di concentrarsi su determinate funzionalità o sezioni dell'applicazione. Il merge di tali branch nel branch principale (main) è stato eseguito solo dopo una revisione accurata, garantendo l'integrità del codice.

In questo modo, il nostro approccio si avvicina al mondo Agile guidato da una combinazione di ideazione, progettazione iterativa, agilità nella risposta ai requisiti e una gestione collaborativa del codice attraverso strumenti avanzati come GitHub. Questa metodologia ha contribuito al successo del progetto, fornendo una base solida per la creazione di un'applicazione Social coinvolgente e ben strutturata.

Struttura dell'Applicazione: Il progetto è organizzato in diverse sezioni, ognuna con un compito specifico:

1. **Pagina di Login e Registrazione:**
    - Accesso tramite account Google con servizi di autenticazione implementati.
    - Registrazione dell'utente attraverso l'intercettazione e la memorizzazione dei dati.
2. **Homepage:**
    - Visualizzazione dei post degli utenti seguiti, con possibilità di commentare i post.
    - Creazione e visualizzazione dei post personali.

3. **Profile Page:**
    - Lista dei post con possibilità di aggiungere commenti, modificarli o eliminarli.
    - Visualizzazione e gestione della lista di amici, in particolare follow e unfollow.
    - Aggiunta dei dati del proprio animale con possibilità di visualizzarli.
4. **Cerca Amici:**
    - Ricerca e aggiunta di nuovi amici tra quelli suggeriti.
5. **Gestione Amici:**
    - Possibilità di non seguire o eliminare un amico dalla propria lista.

Conclusioni: Il progetto ha seguito un approccio di sviluppo con una netta separazione delle responsabilità. Angular si è occupato del frontend, Fastify e Node.js del backend, MongoDB del database, con Postman impiegato per il testing delle API. Le funzionalità implementate consentono agli utenti di gestire i propri post, connettersi con gli amici e personalizzare il loro profilo, offrendo un'esperienza interattiva e coinvolgente. Possibili sviluppi futuri potrebbero concentrarsi sulla sicurezza, sull'ottimizzazione delle prestazioni e sull'implementazione di nuove funzionalità avanzate. La separazione delle responsabilità favorisce la modularità, la manutenibilità e la scalabilità del sistema.