TECNOLOGIE WEB
progetto Selfie

A.A. 2023-24

Istruzioni per compilare e usare l'applicazione

1)Aprire cartella SELFIE-main dal terminale
2)andare nella cartella backend sempre da terminale con il comando: "*cd backend"* partendo dalla cartella SELFIE-main
2-a)eseguire il comando "*npm install*", per installare le varie dipendenze per la parte backend
3)andare nella cartella frontend in un terminale differente con il comando "*cd frontend*" partendo dalla cartella SELFIE-main
3-a)eseguire il comando "*npm install*", per installare le varie dipendenze della parte di frontend
4)nel terminale relativo al backend eseguire il comando "*npm start*"
5)nell'altro terminale, relativo al frontend, eseguire il comando "*ng serve*" o in alternativa sempre "*npm start*"
6)aprire il proprio browser(nel nostro caso abbiamo utilizzato chrome) e andare al seguente url http://localhost:4200/ e qui sarà possibile utilizzare l'applicazione
7)Per chiudere l'applicazione, terminare i processi nei due terminali in cui sono stati avviati i servizi

Per testare l'applicazione è possibile utilizzare i dati del seguente utente:

username: mario12

password: 091024

o creare nuovi account.

Architettura del progetto

- Backend(Nodejs):
    - Controllers
    - Models
    - Routes
    - Data
- Frontend(Angular):
    - Components
    - Services
    - Guards
    - Models

Il progetto è suddiviso in due componenti principali: il backend e il frontend, che collaborano per fornire un'applicazione completa e funzionale. Il frontend è responsabile dell'interfaccia utente e delle interazioni grafiche, mentre il backend gestisce la logica applicativa, la persistenza dei dati e le API che servono il frontend.

Backend

Il lato backend dell'applicazione è stato sviluppato con node v20 ed è stato organizzato in diverse cartelle, ognuna con un ruolo specifico nella gestione delle operazioni e della persistenza dei dati.
La struttura principale include:

- Controllers:
Questa cartella contiene i vari controller, che rappresentano i metodi attraverso i quali si può interagire con i dati presenti nei file JSON. I controller gestiscono le operazioni di creazione, lettura, aggiornamento e cancellazione (CRUD) per entità come ad esempio utenti, le note ecc...
Ad esempio, nel file userController ci sono metodi per la gestione della registrazione e autenticazione, mentre nel controller delle note ci sono metodi per aggiungere, modificare o eliminare le note.

- Data:
La cartella `data` contiene i file JSON , che fungono da database locale per la persistenza dei dati. Questi file memorizzano le informazioni necessarie per l'applicazione, ad esempio gli utenti. Inoltre, contiene una sottocartella *music* , dove vengono archiviate le tracce mp3 caricate dagli utenti e che possono essere riprodotte nella schermata Pomodoro Timer .

Attraverso l'interfaccia grafica, gli utenti possono caricare nuove tracce, che vengono automaticamente salvate in questa cartella.

Esempi di file JSON:
- `users.json`: Memorizza la lista degli utenti registrati.
- `userNotes.json`: Memorizza le note create dagli utenti.

- Models:
La cartella `models` contiene le classi che rappresentano le strutture dati utilizzate nell'applicazione. Ogni modello rappresenta un'entità del sistema, come un utente o una nota. Queste classi definiscono le proprietà e i metodi associati a ciascuna entità.

Ad esempio:

- User: Definisce un utente del sistema con campi come `username`, `password` ed `email`.
- userNote: Definisce una nota con proprietà come `title`, `content`, e `date`. 'title', 'content', 'categories', 'createdAt', 'updatedAt', 'authorUsername', 'noteColor'

- Routes:
La cartella `routes` contiene le definizioni delle rotte API che collegano il frontend con il backend. Ogni rotta gestisce una specifica richiesta HTTP (GET, POST, PUT, DELETE) e richiama il metodo appropriato dal controller per eseguire l'operazione richiesta. Le rotte facilitano l'accesso ai dati e alle funzionalità dell'applicazione dal lato frontend.

Esempi di percorsi API:

- /users: con chiamata GET si riceve la lista di tutti gli utenti della piattaforma, invece con una chiamata POST si registra un utente nella piattaforma.

Frontend

Il frontend è stato sviluppato utilizzando Angular 18 ed è responsabile dell'interfaccia grafica e dell'interazione con l'utente. L'architettura del frontend è organizzata in diverse cartelle chiave che ospitano componenti, servizi e modelli. Il frontend comunica con il backend attraverso API per gestire la visualizzazione dei dati e l'interazione con l'utente.

- Components:
La cartella `components` contiene i vari componenti che costituiscono l'interfaccia utente dell'applicazione. Ogni componente rappresenta una sezione specifica del frontend, ad esempio:

- NavBar: Include gli elementi della navigazione, che consentono agli utenti di cambiare pagina.
- Ogni pagina dell'applicazione è rappresentata da un componente dedicato, come il Pomodoro Timer , le note e ogni componente è associato a una specifica funzione grafica.

- Guards:

La cartella `guards` contiene delle guardie che proteggono l'accesso alle pagine dell'applicazione. Questi file hanno il compito di controllare che l'utente abbia effettuato l'autenticazione prima di effettuare l'accesso a pagine riservate, come quelle relative alle note personali.

Ad esempio, se un utente non ha effettuato il login, le guardie impediscono l'accesso alla pagina delle note, anche se viene modificato l'url dall'utente stesso.

- Models:

La cartella `models` nel frontend è simile a quella del backend e contiene le classi che rappresentano le strutture dati utilizzate dall'interfaccia utente. Questi modelli rispecchiano le entità del backend (come User e userNote) e vengono utilizzati per tipizzare i dati ricevuti o inviati tramite le API.

- Services:

I servizi nella cartella `services` contengono le logiche per le chiamate HTTP al backend. Questi servizi permettono al frontend di comunicare con il backend per recuperare o inviare dati. Ogni servizio è associato a un'entità specifica, come UserService per la gestione degli utenti o NoteService per la gestione delle note.

I servizi sono essenziali per:
- Ottenere le informazioni dell'utente dal backend.
- Creare, modificare o eliminare note.
- Caricare e riprodurre tracce audio.

Conclusione

Il progetto è strutturato in modo da separare le responsabilità tra backend e frontend, garantendo una comunicazione chiara e flessibile attraverso le API. Il backend si occupa della gestione dei dati e della loro persistenza, mentre il frontend si concentra sull'interazione utente e sulla presentazione dei dati. Grazie alla struttura modulare di Angular e alla suddivisione in componenti, servizi e modelli, l'applicazione è facile da mantenere, estendere e aggiornare.