Nome progetto:

Tripply

Gruppo G42 - Deliverables 4

Scopo del documento

Il documento fornisce le informazioni essenziali per la realizzazione dell'applicazione Tripply. Include una dettagliata descrizione dei flussi utente, delineando le interazioni e le fasi coinvolte per gli utenti. A seguire vegono presenttate le API necessarie per la creazione, modifica e visualizzazione degli itinerari.

User Flows

Nel seguente capitolo vengono riportati gli Users Flows che hanno lo scopo di rappresentare come un utente interagirà con il sito web.
Sono stati individuati tre tipi di utenti: Utente non autenticato, utente autenticato e amministratore.

Utente non autenticato

Utente autenticato

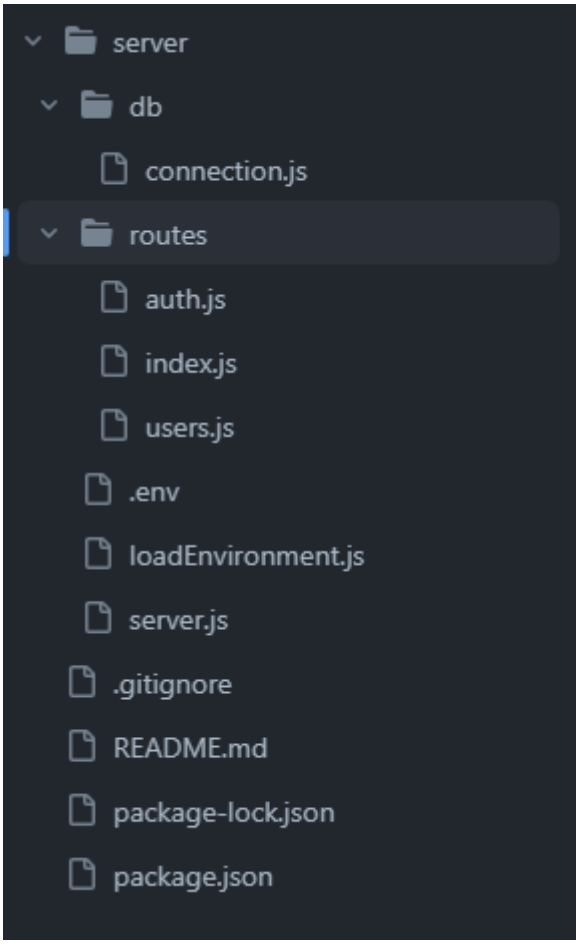
Amministratore

Application Implementation and Document

Nel seguente capitolo vengono presentati le varie features che verranno implementate nell'applicazione.

Tripply è stato sviluppato utilizzando Express.js versione 4.18.2, framework leggero e flessibile per lo sviluppo di applicazioni web e API.

Project Structure



Project Dependencies

Project DataBase

Project APIs

Resouce Models

Sviluppo API

```
// Create a new itinerary
Daniele Ye
router.post( path: '/createItinerary', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
    const itinerary : ReqBody = req.body;
    const result : InsertOneResult<Document> = await dbtest.collection( name: "Itinerario").insertOne(itinerary);
    res.send(result);
});

// Get all user itineraries
Daniele Ye
router.get( path: '/getUserItineraries/', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
    let userId : number = parseInt(req.query.userId);
    try {
        const result : (WithId<...>)[] = await dbtest.collection( name: "Itinerario").find( filter: { "_userId": userId }).toArray();
        res.send(result);
    } catch (error) {
        console.error("Error fetching itineraries:", error);
        res.status( code: 500).send( body: "Internal Server Error");
    }
});

// Get user info
Daniele Ye
router.get ( path: '/getUserInfo/', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
    let userId : number = parseInt(req.query.userId);
    try {
        const result : Document & {...} = await dbtest.collection( name: "Utente").findOne({ "_id": userId });
        res.send(result);
    } catch (error) {
        console.error("Error fetching user info:", error);
        res.status( code: 500).send( body: "Internal Server Error");
    }
});
```