API Endpoints

Setup

Il codice seguente va inserito dentro index.js, uno per ogni file di route che viene creato.
Il primo argomento è il prefisso di tutte le route, il secondo è il file che le contiene.

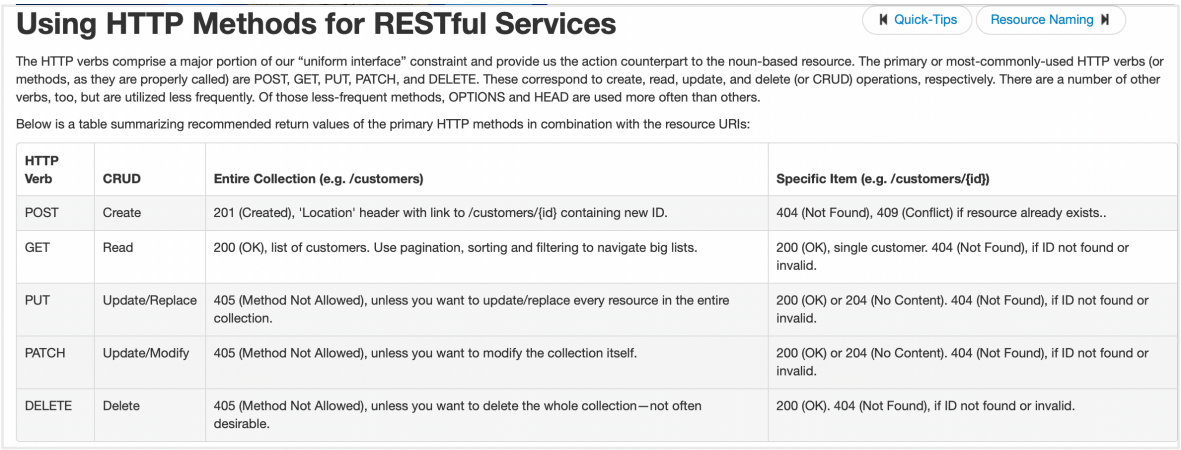
app.use('<prefisso>', require('./routes/<nome-file>'))

Le route nei vari file vanno poi definite senza prefisso; quello viene aggiunto automaticamente una volta che viene inserita la riga di codice qui sopra.
Esempio (preso dal file "community-routes.js"):

router.get('/insurance', async (req, res, next) => ...

Risorse utili per costruire una api restful

Ogni chiamata all'api poi ritorna anche uno dei codici dell'immagine sotto in base a cosa succede alla richiesta



Using HTTP Methods for RESTful Services

Quick-Tips | Resource Naming

The HTTP verbs comprise a major portion of our "uniform interface" constraint and provide us the action counterpart to the noun-based resource. The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently. Of those less-frequent methods, OPTIONS and HEAD are used more often than others.

Below is a table summarizing recommended return values of the primary HTTP methods in combination with the resource URIs:

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

https://www.restapitutorial.com/lessons/httpmethods.html

Ho trovato questo per salvare immagine in mongoDB
https://stackoverflow.com/questions/29780733/store-an-image-in-mongodb-using-node-js-express-and-mongoose

Questo invece approccio per salvare immagini tramite API.
Discutono entrambi delle differenze tra usare multipart/form-data e inviare immagine in base64 su json.
https://stackoverflow.com/questions/33279153/rest-api-file-ie-images-processing-best-practices

https://nimesha-dilini.medium.com/send-image-files-in-an-api-post-request-aa1af1c4a7fb

Annunci
Prefisso: *"/api/family-market/postings"*

GET "/groups/:group_id"
Restituisce tutti gli annunci postati appartenenti al gruppo specificato.

Esempio chiamata:
GET "*/api/family-market/postings/groups/1"*

Esempio di risposta dell'api:
[
{
id: "1"
user_id: "1"
group_id: "1"
name: "Annuncio1"
category: "Categoria1"
description: "Descrizione1"
foto: <byte della foto convertiti in base64>
type: "donation"
contact: {
email: "ciao@gmail.com"
place: "Mestre"
phone_number: "3317776655"
}
},
{
id: "2"
user_id: "2"
group_id: "1"
name: "Annuncio2"
category: "Categoria2"
description: "Descrizione2"
foto: <byte della foto convertiti in base64>
type: "donation"
contact: {
email: "ciao2@gmail.com"
place: "Mestre"
phone_number: "3327776655"
}
},
...
]

GET "/:id"
Restituisce l'annuncio il cui id è quello specificato.

Esempio chiamata:
GET *"/api/family-market/postings/1"*

Esempio di risposta dell'api:
{
id: "1"
user_id: "1"
group_id: "1"
name: "Annuncio1"
category: "Categoria1"
description: "Descrizione1"
foto: <byte della foto convertiti in base64>
type: "donation"
contact: {
email: "ciao@gmail.com"
place: "Mestre"
phone_number: "3317776655"
}
}

POST "/"
Crea un annuncio e risponde ritornando l'annuncio appena creato

Esempio chiamata:
POST *"/api/family-market/postings"*

Esempio di body della richiesta:
{
user_id: "1"
group_id: "1"
name: "Annuncio1"
category: "Categoria1"
description: "Descrizione1"
foto: <byte della foto convertiti in base64>
type: "donation"
contact: {
email: "ciao@gmail.com"
place: "Mestre"
phone_number: "3317776655"
}
}

Esempio di risposta dell'api:
{
id: "1"

user_id: "1"
group_id: "1"
name: “Annuncio1”
category: “Categoria1”
description: “Descrizione1”
foto: <byte della foto convertiti in base64>
type: “donation”
contact: {
email: “ciao@gmail.com”
place: “Mestre”
phone_number: “3317776655”
}
}

PATCH “/:id”
Modifica un annuncio esistente, il cui id è quello specificato.

Esempio chiamata:
PATCH *“/api/family-market/postings/1”*

Esempio di body della richiesta:
{
name: “Annuncio2”
category: “Categoria2”
description: “Descrizione2”
foto: <byte della foto convertiti in base64>
type: “loan”
contact: {
email: “ciao@gmail.com”
place: “Mestre”
phone_number: “3317776655”
}
}

Risposta dell’api :
{
id: "1"
user_id: "1"
group_id: "1"
name: “Annuncio2”
category: “Categoria2”
description: “Descrizione2”
foto: <byte della foto convertiti in base64>
type: “donazione”
contact: {
email: “ciao@gmail.com”

```
                    place: “Mestre”
                    phone_number: “3317776655”
                }
}
```

DELETE “/:id”
Rimuove un annuncio esistente, il cui id è quello specificato.

Esempio chiamata:
DELETE “*/api/family-market/postings/1”*

Users (estensione)

Anche se c’è già un file con le route relative allo user, per quelle che aggiungiamo noi probabilmente ha senso metterle su un file a parte ed assegnare loro un prefisso differente.

Prefisso: “/api/family-market/users/“

GET “/:id/postings”
Ottiene tutti gli annunci postati (e correntemente attivi) dell’utente con l’id specificato, raggruppati per gruppo di appartenenza.

Esempio chiamata:
GET “*/api/family-market/users/1/postings”*

Esempio di risposta dell’api:

```
[
  {
    group_info:
      {
        id: "1"
        name: “Gruppo1”
      }
    postings: [
          {
            id: "1"
            user_id: "1"
            group_id: "1"
            name: “Annuncio1”
            category: “Categoria1”
            description: “Descrizione1”
            foto: <byte della foto convertiti in base64>
            type: “donation”
            contact: {
```

```
                        email: “ciao@gmail.com”
                        place: “Mestre”
                        phone_number: “3317776655”
                    }
            },
            ...
            ]
    },
{
    group_info:
        {
            id: "2"
            name: “Gruppo22”
        }
    postings: [
            {
                id: "1"
                user_id: "1"
                group_id: "2"
                name: “Annuncio6”
                category: “Categoria1”
                description: “Descrizione1”
                foto: <byte della foto convertiti in base64>
                type: “donation”
                contact: {
                        email: “ciao@gmail.com”
                        place: “Mestre”
                        phone_number: “3317776655”
                    }
            },
            ...
            ]
    },
    ...
]
```

GET “/:id/favourites”
Restituisce la lista di annunci preferiti dell’utente specificato.

Esempio chiamata:
GET *“/api/family-market/users/1/favourites”*

Esempio di risposta dell’api:

[
{
id: "1"
user_id: "1"
group_id: "1"
name: “Annuncio1”
category: “Categoria1”
description: “Descrizione1”
foto: <byte della foto convertiti in base64>
type: “donation”
contact: {
email: “ciao@gmail.com”
place: “Mestre”
phone_number: “3317776655”
}
},
{
id: “3”
user_id: "1"
group_id: "1"
name: “Annuncio3”
category: “Categoria3”
description: “Descrizione3”
foto: <byte della foto convertiti in base64>
type: “donation”
contact: {
email: “ciao2@gmail.com”
place: “Mestre”
phone_number: “3327776655”
}
},
...
]

PUT “/:id/favourites”
Modifica il campo “favourites” (contenente la lista dei preferiti) dell’utente specificato e ritorna il campo aggiornato.

Esempio chiamata:
PUT *“/api/family-market/users/1/favourites”*

Esempio di body della richiesta:
{
favourites: [1, 6, 22, 8, 184]
}

Esempio di risposta dell'api:

```
{
      favourites: [1, 6, 22, 8, 184]
}
```