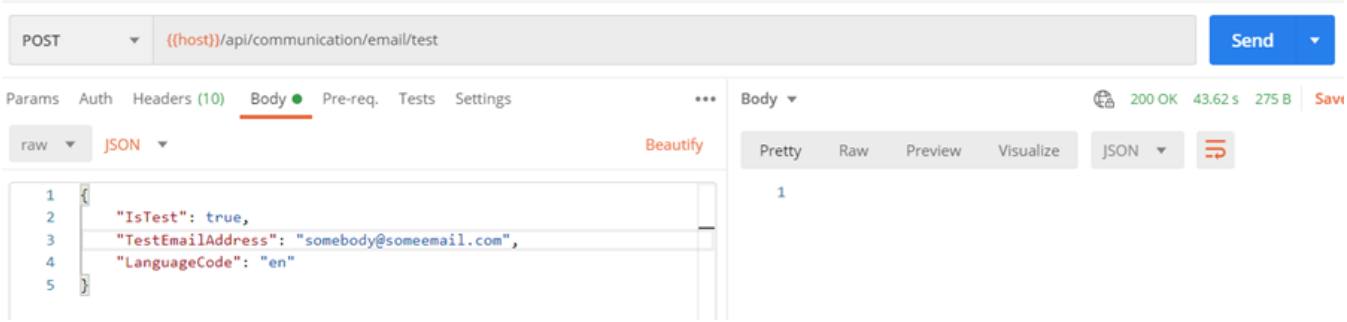
1. REST API MUST ACCEPT AND RESPOND WITH JSON



2. GO WITH ERROR STATUS CODES

```
router.get('/api/profile/:id', function(req,res,next){
    Users.findOne({id:req.params.id}, function(error,user){
        if(error){
            return res.status(500).json(error)
        }
        if(!user){
            return res.status(404).json({"error":"User ID not found."})
        } else {
            res.send(user)
        }
    })
})
```

3. DON'T USE VERBS IN URLS

Correct way

```
GET /books/123
DELETE /books/123
POST /books
PUT /books/123
PATCH /books/123
```

Incorrect way

```
GET /addBook123
GET /DeleteBooks/123
POST /DeleteAllBooks
POST /books/123/delete
```

4. USE PLURAL NOUNS TO NAME A COLLECTION

Do

`GET /cars/123`
`POST /cars`
`GET /cars`

Don't

`GET /car/123`

`POST /car`
`GET /car`

5. WELL COMPILED DOCUMENTATION

"Pasted image 20221205080532.png" is not created yet. Click to create.

6. RETURN ERROR DETAILS IN THE RESPONSE BODY

```
{
        "error": "Invalid payoad.",
        "detail": { "surname": "This field is required." }
}
```

7. USE RESOURCE NESTING

`/users // list all users`

`/users/123 // specific user`

`/users/123/orders // list of orders that belong to a specific user`

`/users/123/orders/0001 // specific order of a specific users order list`

8. Filtering, sorting, paging, and field selection

Filtering:

```
GET /users?country=ID
GET /users?creation_date=2019-11-11
```

Sorting:

```
GET /users?sort=birthdate_date:asc
GET /users?sort=birthdate_date:desc
```

Paging:

```
GET /users?limit=100
GET /users?offset=2
```

All together:

```
GET /users?country=USA&creation_date=2019-11-
11&sort=birthdate_date:desc&limit=100&offset=2
```

Fields:

```
GET /users/123?fields=username,email (for one specific user)
GET /users?fields=username,email (for a full list of users)
```

9. Versioning

Examples of endpoint URI versioning:

- https://us6.api.mailchimp.com/3.0/ (major + minor version indication)
- https://api.stripe.com/v1/ (major version indication only)
- https://developer.github.com/v3/ (major version indication only)
- https://api.twilio.com/2010-04-01/ (date based indication)

10. USE SSL/TLS

11. SECURE YOUR API

Your API must have an HTTP Strict Transport Security (HSTS) policy. Up next, you should secure your network from middle man attacks, protocol downgrade attacks, session hijacking, etc., Just use all the relevant security standards to the security of your API.