

# Sistemes Oberts

## Practica 1

Desenvolupament d'una API REST

Raul Martin Morales  
Elyas El Kouissi Charkaoui  
Curs 2024/2025

## **Index:**

1. Introducció
2. Estructura de la pràctica
3. Decisions de disseny
4. Jocs de proves realitzades
5. Conclusions
6. Manual d'instal·lació

## 1. Introducció

Aquesta primera pràctica, ens enfoquem en el desenvolupament d'una API web dissenyada per gestionar la nostra base de dades d'articles amb usuaris. L'objectiu principal de la pràctica és crear una API RESTful que ens permet fer una gestió bàsica d'articles com afegir, consultar, eliminar i modificar. Per altra banda hem de gestionar també el llistar i consultar usuaris individualment, controlant també la seguretat de les seves credencials, addicionalment una autenticació bàsica per a poder accedir a funcionalitats que requereixin de autenticació.

També volem familiaritzar-nos amb l'ús de formats de dades JSON i els codis HTTP segons el resultat de les operacions.

## 2. Estructura de la pràctica

Els principal desenvolupament que hem fet per dur a terme la pràctica i poder gestionar aquesta base de dades s'enmarca en el `src/java`, que conté les classes i el codi principal per gestionar la base de dades.

`src/java/model/entities`: On es defineixen les entitats JPA per representar els articles, usuaris i topics.

La classe `Article` que representa un article amb camps com títol, autor, visualitzacions, resum, image, data de publicació, topics i privacitat.

La classe `Usuari` que representa un usuari amb camps com nom, correu i edat.

La classe `Topic` guarda solament el nom, i el id generat automàticament com en totes les altres classes.

`src/java/service`: Té les implementacions dels serveis que ens permeten gestionar els articles i usuaris.

`ArticleService` es on hem implementat les operacions com GET, POST i DELETE, per poder llistar els articles tant per autor, com per topic o id, afegir articles i eliminar-lo si ets el autor.

`UsuariService` es on hem implementat les operacions de GET I PUT dels Usuaris segons diferents condicions.

`src/java/authn`: Tenim les classes que ens permeten controlar la autenticació de l'usuari.

`Credentials` on guardem el username i contrasenya del usuari que li permetrà autenticar-se i adquirir diferents permisos.

`WebPages/`: Es on tenim el arxiu `install.jsp` on ens encarreguem de fer els inserts manualment.

### 3. Decisions de disseny

Durant el desenvolupament del projecte hem hagut de prendre diferents decisions de disseny en segons quina funció per a que es comportin segons la nostra decissió.

Pels Articles:

Al GET `/rest/api/v1/article?topic=${topic}&author=${author}` hem decidit que es permet especificar fins 2 topics, per altra banda, si no hi ha resultats retornem una resposta `NOT_FOUND`. Es cercaran articles que contenguin algun topic dels dos possibles a especificar. I com no s'especifica per aquesta consulta, no es farà la gestió d'autenticació del usuari.

Al GET `/rest/api/v1/article/${id}` primerament fem la autenticació del usuari i segons el resultat podrem retornar un article depenent si es privat o no. I si no es troba el article retornem un `204 No Content` i si existeix pero no té permisos també.

Al DELETE `/rest/api/v1/article/${id}` hem fet una funció `validarAutor` que ens retorna el autor del article amb aquesta id i si es igual Usuari doncs podrà eliminar-lo, enviarà `401 Unauthorized` si no està autoritzat a eliminar-lo.

Al POST `/rest/api/v1/article` validem que el article ha de tenir exactament 2 tooics i que aquests existeixen a la base de dades. També associem automàticamente el usuari que fa el post com el nom del autor. Si es crea correctament retorna `201 Created`.

Pels Usuaris:

Al GET `/rest/api/v1/customer` hem decidit que per obtenir el link de l'últim article que ha publicat aquest usuari, de la llista dels seus articles obtenim el id de l'últim en aquesta llista.

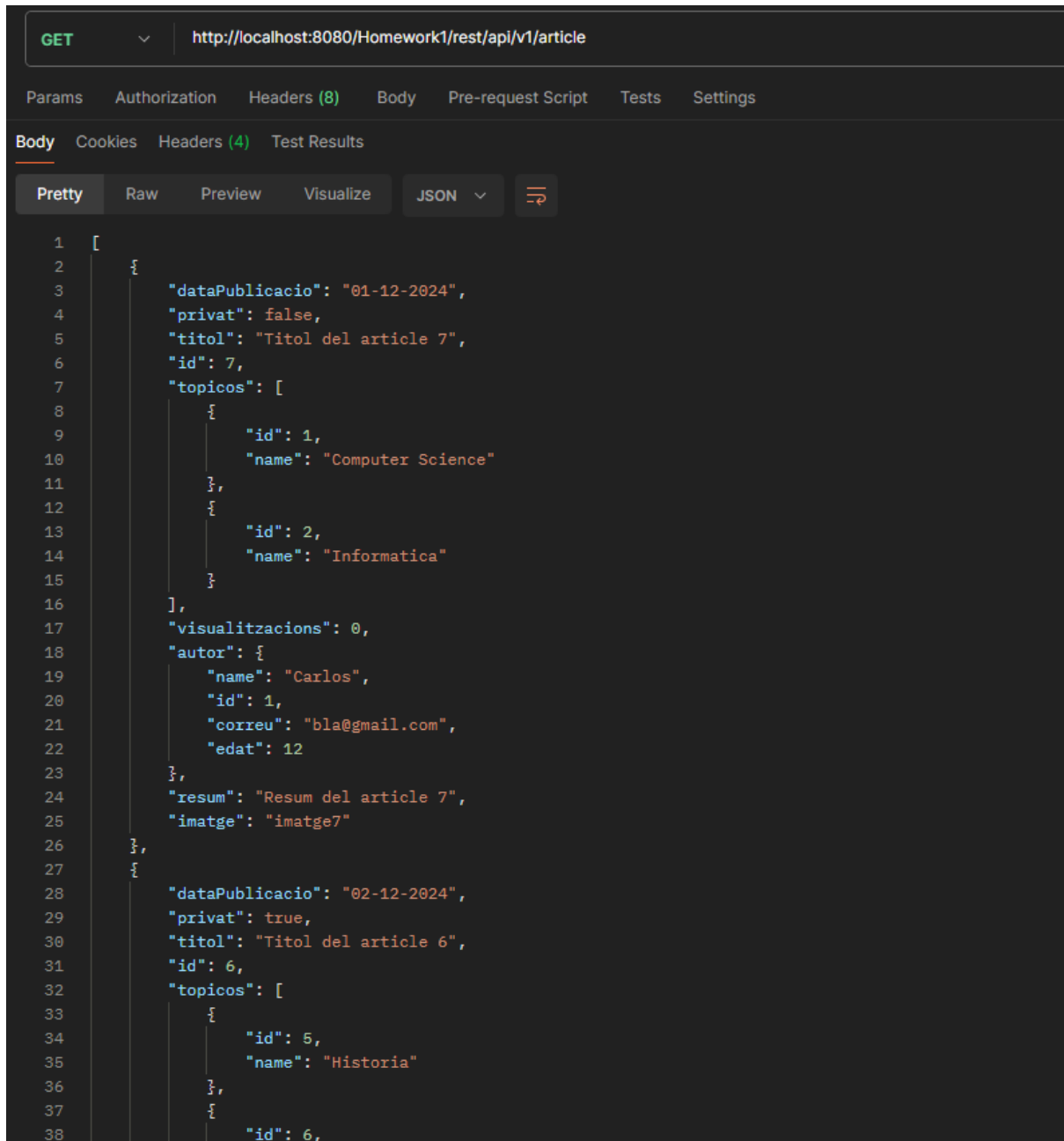
Al GET `/rest/api/v1/customer/${id}` com en totes les funcions hem ocultat els credencials dels usuaris. Si no es troba el usuari retornem un `204 No Content`.

Al PUT `/rest/api/v1/customer/${id}` per actualitzar les dades d'un usuari has de ser el propi usuari el que fa els canvis, aleshores has d'estar autenticat.

## 4. Joc de proves realitzats

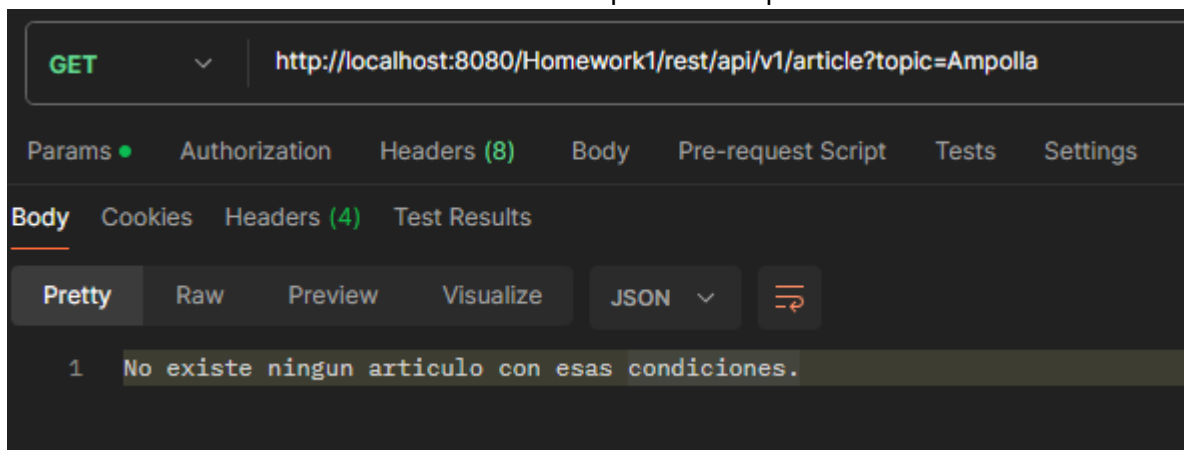
- Article
  - GET de tots els articles
  - GET de tots els articles especificant dos tópics correctes i un autor existent
  - GET de tots els articles especificant tópics inexistents
  - GET de tots els articles especificant un autor inexistent
  - GET augmenten les visualitzacions i es veun de major a menor
  - GET d'un article privat segons el seu id estant autenticat
  - GET d'un article privat segons el seu id sense estar autenticat
  - DELETE d'un article sent l'autor
  - DELETE d'un article sense ser l'autor
  - DELETE d'un article que no existeix
  - POST d'un article vàlid
  - POST d'un article amb tópics inexistents
  - POST d'un article sense estar autenticat
- Usuari
  - GET de tots els usuaris
  - POST article d'un usuari
  - GET de tots els usuaris (comprovar que s'actualitza el link a l'últim article)
  - GET d'un usuari segons el seu id
  - GET d'un usuari segons un id inexistent
  - PUT modificar les dades d'un usuari estant autenticat
  - PUT modificar les dades d'un usuari sense autenticar-se
  - PUT modificar el id d'un usuari
  - PUT modificar les dades d'un usuari que no es el que fa la petició

## GET de tots els articles



```
1  [
2    {
3      "dataPublicacio": "01-12-2024",
4      "privat": false,
5      "titol": "Titol del article 7",
6      "id": 7,
7      "temes": [
8        {
9          "id": 1,
10         "nom": "Computer Science"
11       },
12       {
13         "id": 2,
14         "nom": "Informatica"
15       }
16     ],
17     "visualitzacions": 0,
18     "autor": {
19       "nom": "Carlos",
20       "id": 1,
21       "correu": "bla@gmail.com",
22       "edat": 12
23     },
24     "resum": "Resum del article 7",
25     "imatge": "imatge7"
26   },
27   {
28     "dataPublicacio": "02-12-2024",
29     "privat": true,
30     "titol": "Titol del article 6",
31     "id": 6,
32     "temes": [
33       {
34         "id": 5,
35         "nom": "Historia"
36       },
37       {
38         "id": 6,
```

## GET de tots els articles especificant t pics inexistent



```
1  No existe ningun articulo con esas condiciones.
```

GET de tots els articles especificant dos tópics correctes i un autor existent

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/Homework1/rest/api/v1/article?topic=Historia&topic=Letras&author=Elyas`. The response body is displayed in JSON format, showing two articles. The first article (id: 6) has topics 'Historia' and 'Letras' and is authored by 'Elyas' (id: 3). The second article (id: 3) has the topic 'Historia' and is also authored by 'Elyas' (id: 3).

```
1 [
2   {
3     "dataPublicacio": "02-12-2024",
4     "privat": true,
5     "titol": "Titol del article 6",
6     "id": 6,
7     "topicos": [
8       {
9         "id": 5,
10        "name": "Historia"
11      },
12      {
13        "id": 6,
14        "name": "Letras"
15      }
16    ],
17    "visualitzacions": 0,
18    "autor": {
19      "name": "Elyas",
20      "id": 3,
21      "correu": "blablabla@gmail.com",
22      "edat": 79
23    },
24    "resum": "Resum del article 6",
25    "imatge": "imatge6"
26  },
27  {
28    "dataPublicacio": "05-12-2024",
29    "privat": false,
30    "titol": "Titol del article 3",
31    "id": 3,
32    "topicos": [
33      {
34        "id": 5,
35        "name": "Historia"
36      },
37      {
38        "id": 6,
39        "name": "Letras"
40      }
41    ]
42  }
43 ]
```

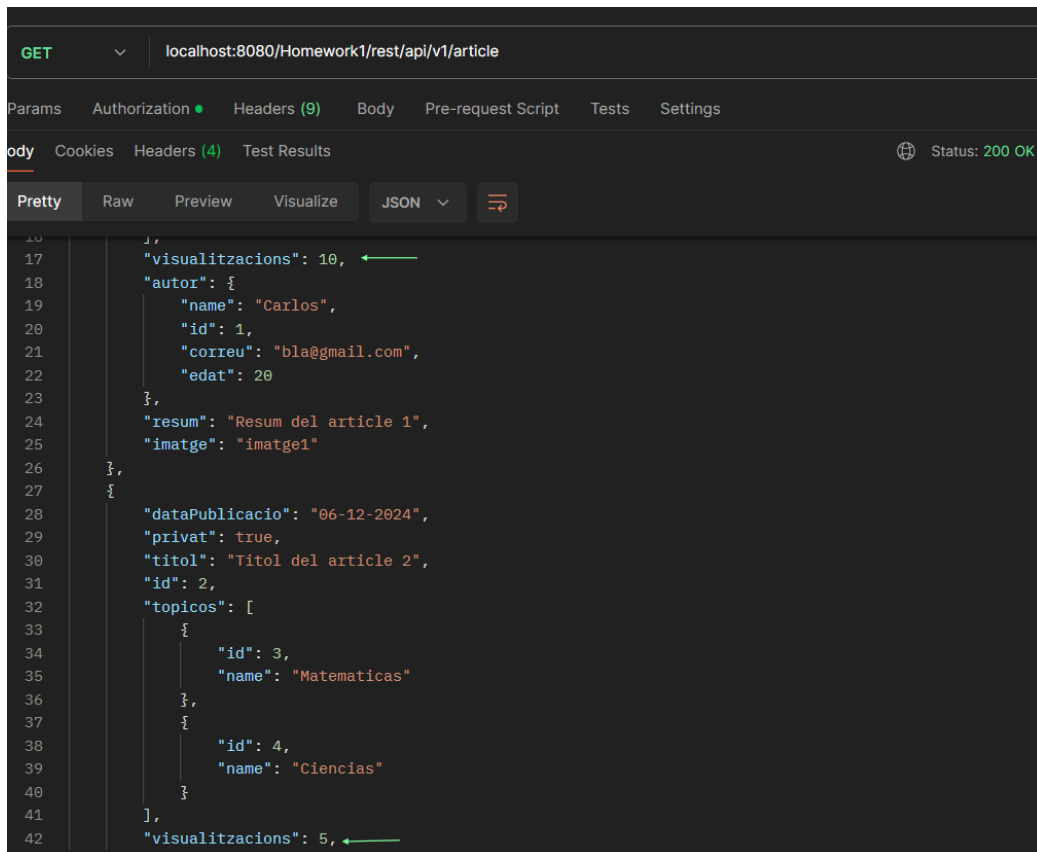
GET de tots els articles especificant un autor inexistent

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/Homework1/rest/api/v1/article?author=JuanAlbertoDeLasMontañas`. The response body is displayed in plain text, indicating that no articles were found for the specified author.

```
1 No existe ningun articulo con esas condiciones.
```

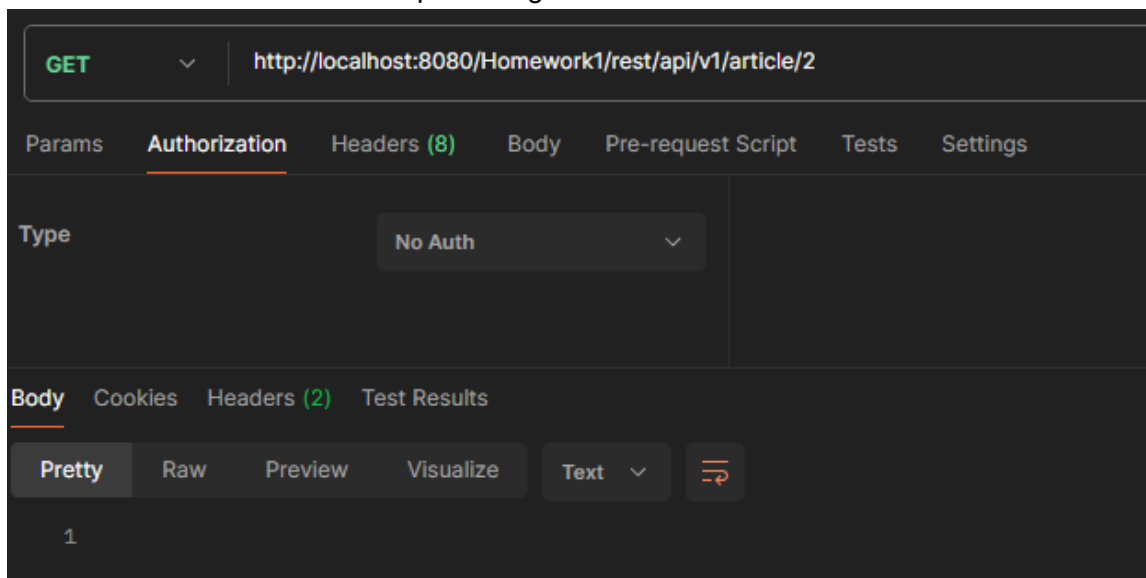


GET augmenten les visualitzacions i es veuen de major a menor



```
16  ],
17    "visualitzacions": 10,
18    "autor": {
19      "name": "Carlos",
20      "id": 1,
21      "correu": "bla@gmail.com",
22      "edat": 20
23    },
24    "resum": "Resum del article 1",
25    "imatge": "imatge1"
26  },
27  {
28    "dataPublicacio": "06-12-2024",
29    "privat": true,
30    "titol": "Titol del article 2",
31    "id": 2,
32    "temes": [
33      {
34        "id": 3,
35        "name": "Matemáticas"
36      },
37      {
38        "id": 4,
39        "name": "Ciencias"
40      }
41    ],
42    "visualitzacions": 5,
```

GET d'un article privat segons el seu id sense estar autenticat



## GET d'un article privat segons el seu id estant autenticat

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/Homework1/rest/api/v1/article/2
- Authorization:** Basic Auth
- Username:** sob
- Password:** sob
- Body:** Pretty view showing a JSON response:

```
1 {
2   "dataPublicacio": "06-12-2024",
3   "privat": true,
4   "titol": "Titol del article 2",
5   "id": 2,
6   "topicos": [
7     {
8       "id": 3,
9       "name": "Matematicas"
10    },
11    {
12      "id": 4,
13      "name": "Ciencias"
14    }
15  ],
16   "visualitzacions": 1,
17   "autor": {
18     "name": "Raul",
19     "id": 2,
20     "correu": "blabla@gmail.com",
21     "edat": 15
22   },
23   "resum": "Resum del article 2",
24   "imatge": "imatge2"
25 }
```

## DELETE d'un article sense ser l'autor

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/Homework1/rest/api/v1/article/2
- Authorization:** Basic Auth
- Username:** sob
- Password:** sob
- Body:** Pretty view showing a text response:

```
1 No eres el autor.
```

## DELETE d'un article sent l'autor

DELETE

http://localhost:8080/Homework1/rest/api/v1/article/2

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Type

Basic Auth

Username

sob2

Password

sob2

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

GET

http://localhost:8080/Homework1/rest/api/v1/article/2

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Type

Basic Auth

Username

sob2

Password

sob2

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body

Cookies

Headers (2)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

DELETE d'un article que no existeix

DELETE

http://localhost:8080/Homework1/rest/api/v1/article/33

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Type

Basic Auth

Username

sob2

Password

sob2

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body

Cookies

Headers (2)

Test Results

Pretty

Raw

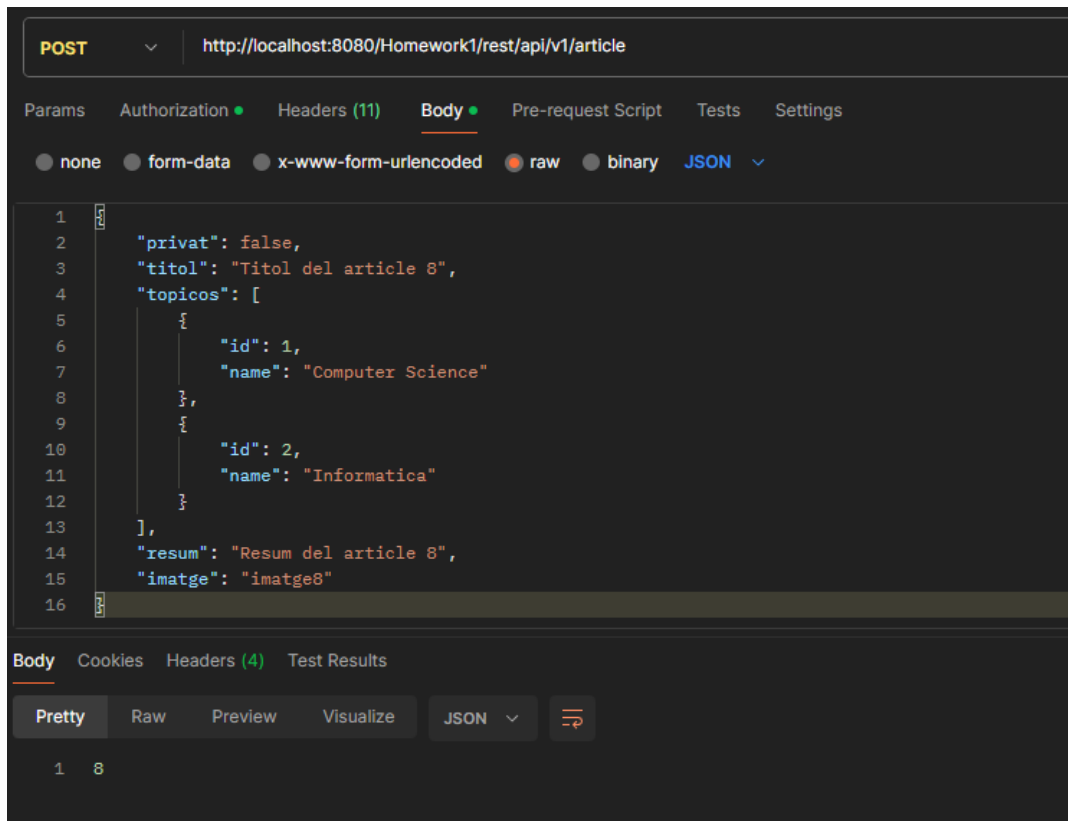
Preview

Visualize

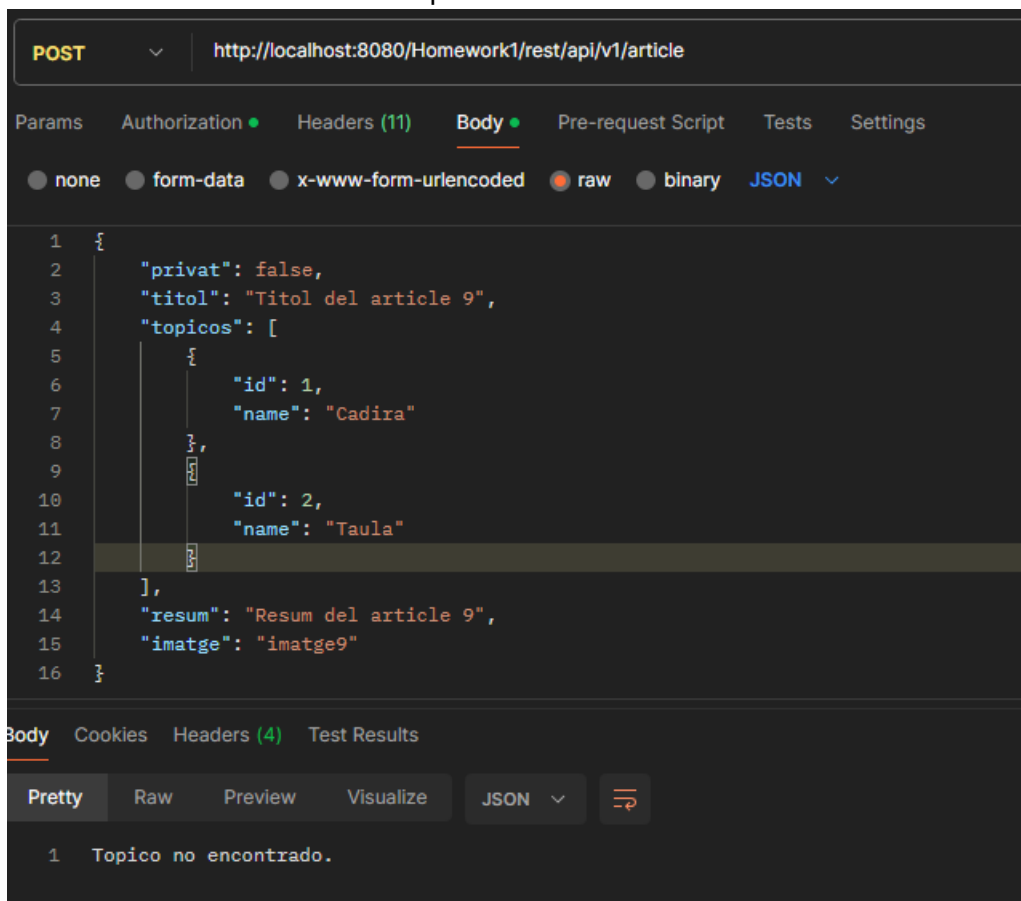
Text

1

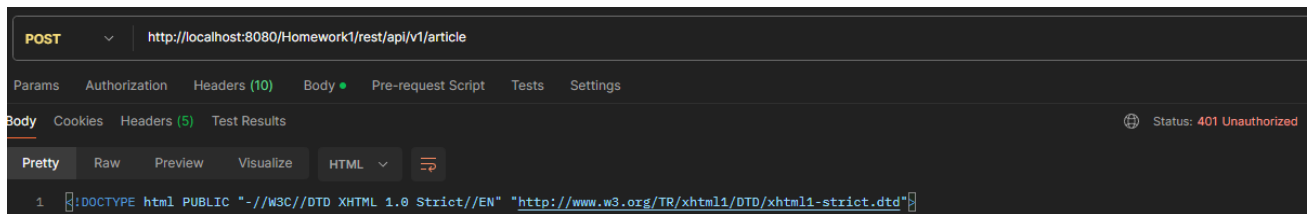
POST d'un article v  lid



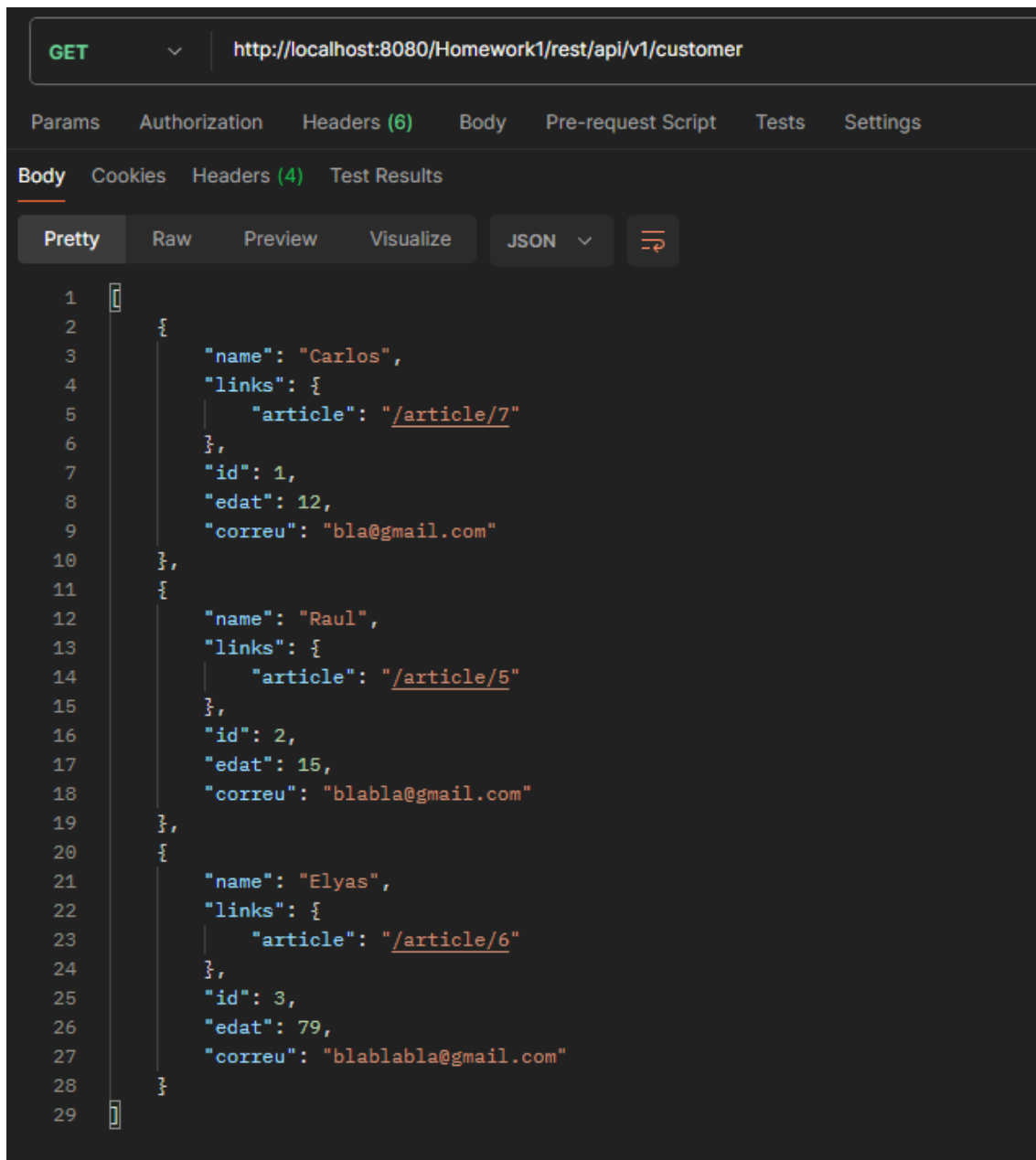
### POST d'un article amb t pics inexistents



### POST d'un article sense estar autenticat



GET de tots els usuaris



POST article d'un usuari i GET de tots els usuaris (per actualitzar últim link)

The screenshot shows a REST client interface with the following components:

- Method:** GET
- URL:** http://localhost:8080/Homework1/rest/api/v1/customer
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with a single header 'Key'.
- Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh icon.
- Response:** A JSON array of three user objects, displayed with line numbers 1 through 29.

```
1  [
2    {
3      "name": "Carlos",
4      "links": {
5        "article": "/article/8"
6      },
7      "id": 1,
8      "edat": 12,
9      "correu": "bla@gmail.com"
10   },
11   {
12     "name": "Raul",
13     "links": {
14       "article": "/article/5"
15     },
16     "id": 2,
17     "edat": 15,
18     "correu": "blabla@gmail.com"
19   },
20   {
21     "name": "Elyas",
22     "links": {
23       "article": "/article/6"
24     },
25     "id": 3,
26     "edat": 79,
27     "correu": "blablabla@gmail.com"
28   }
29 ]
```

GET d'un usuari segons el seu id

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:8080/Homework1/rest/api/v1/customer/1`
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with one column labeled 'Key'.
- Body:** Pretty, Raw, Preview, Visualize, JSON (dropdown), and a refresh icon.
- Response Body (JSON):**

```
1 {
2   "name": "Carlos",
3   "id": 1,
4   "edat": 12,
5   "correu": "bla@gmail.com"
6 }
```

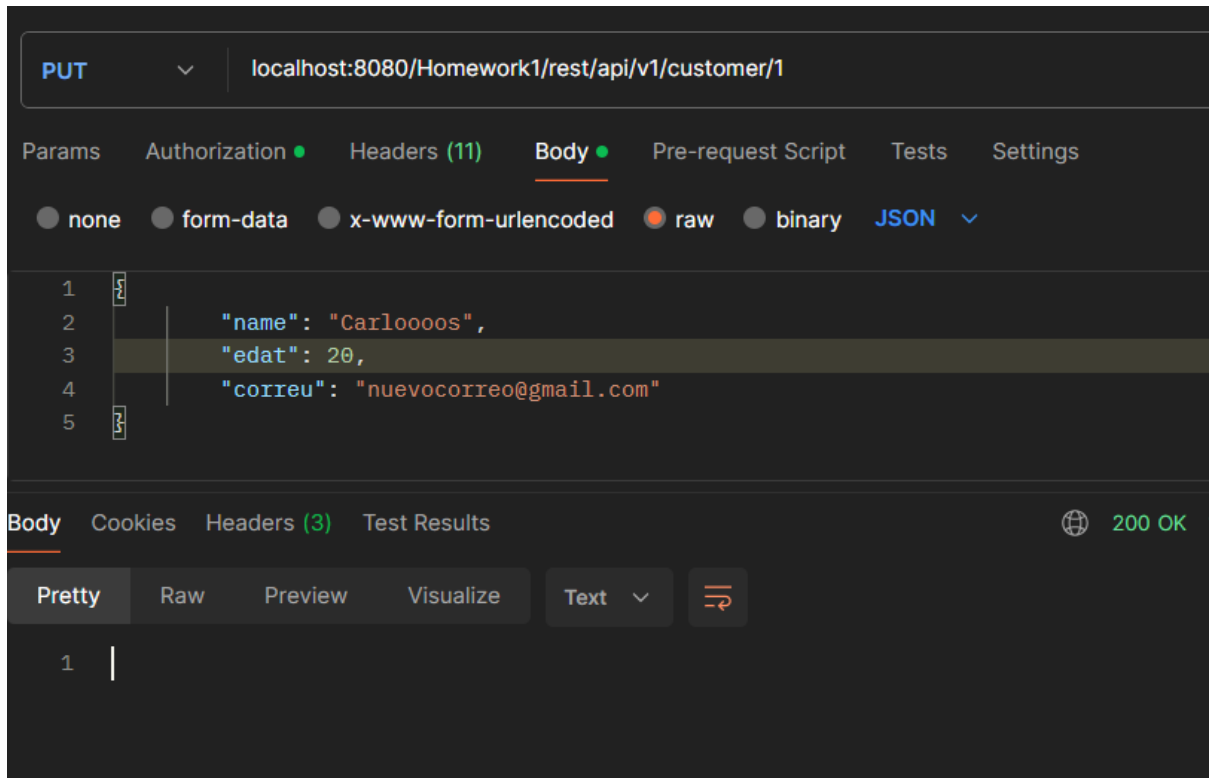
GET d'un usuari segons un id inexistent

The screenshot shows a REST client interface with the following details:

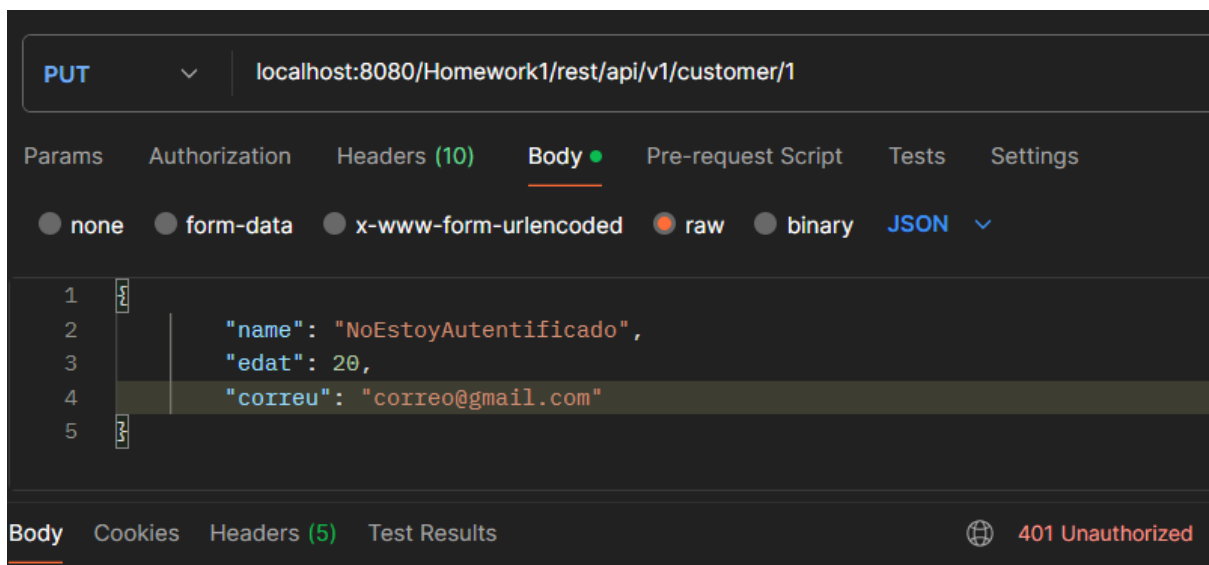
- Method:** GET
- URL:** `http://localhost:8080/Homework1/rest/api/v1/customer/33`
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with one column labeled 'Key'.
- Body:** Pretty, Raw, Preview, Visualize, Text (dropdown), and a refresh icon.
- Response Body (Text):**

```
1
```

PUT modificar les dades d'un usuari estant autenticat

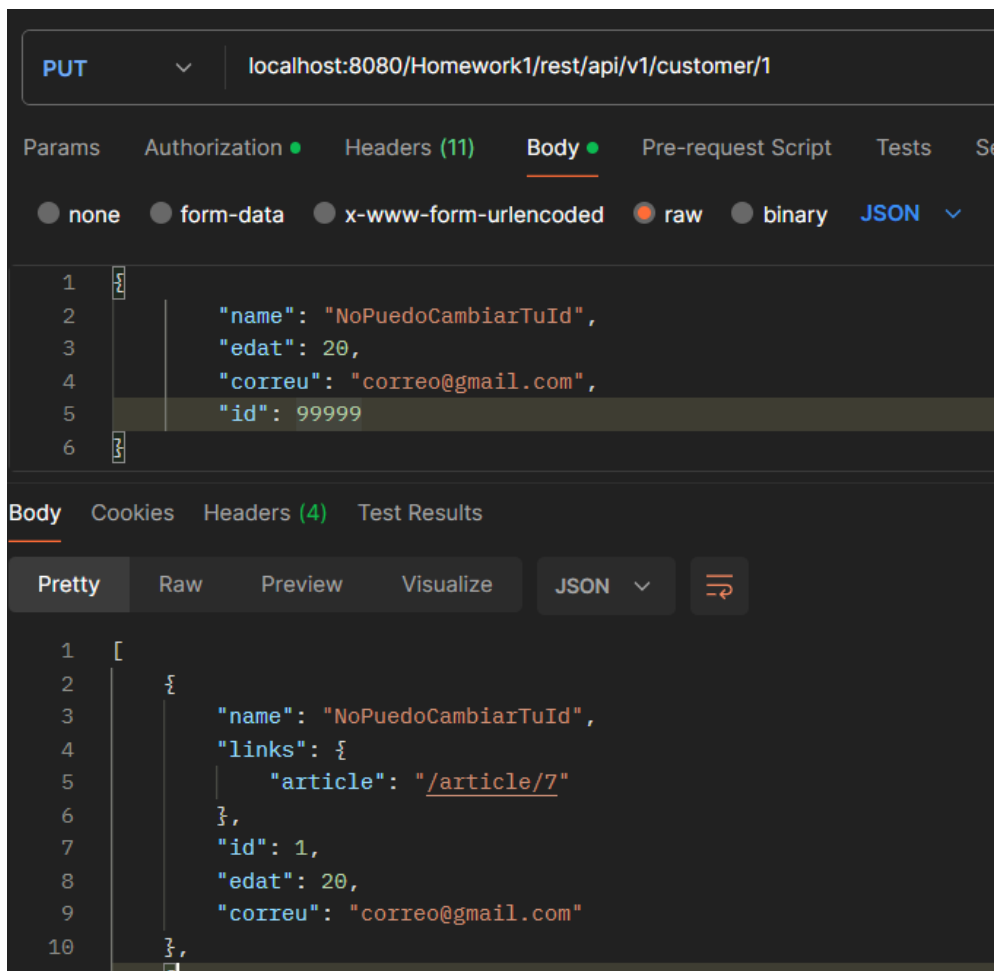


PUT modificar les dades d'un usuari sense autenticar-se

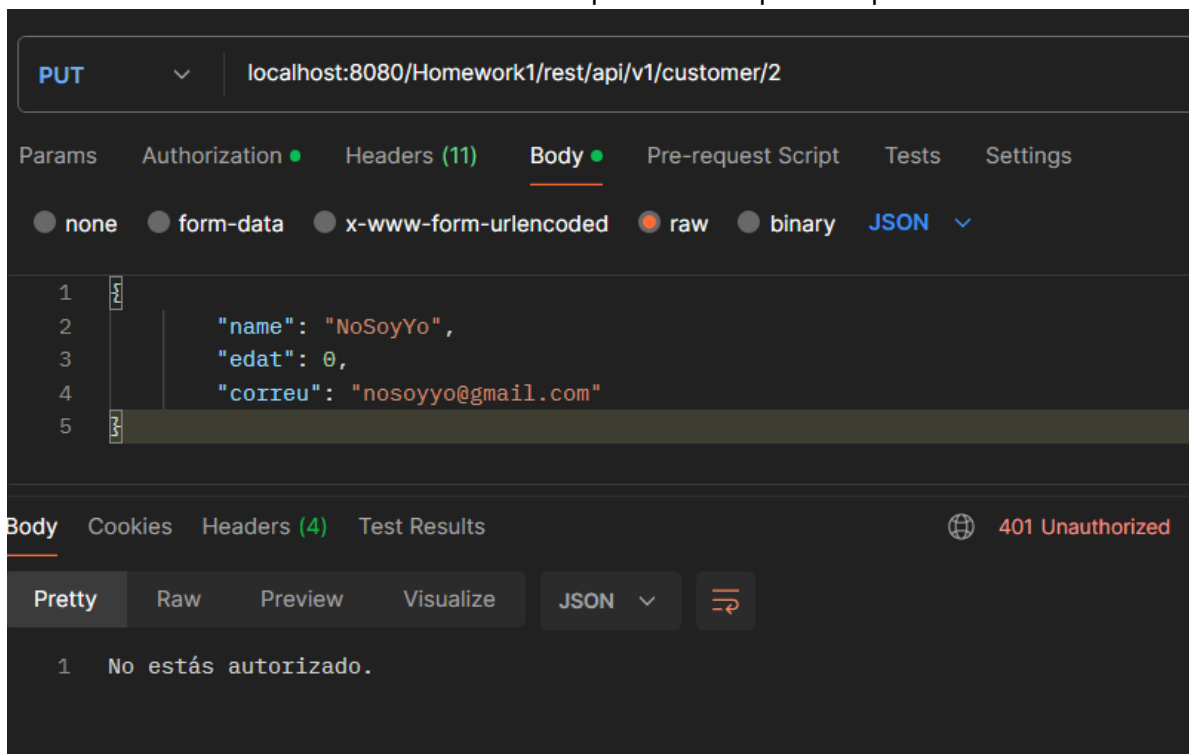


PUT modificar el id d'un usuari





PUT modificar les dades d'un usuari que no es el que fa la petició



## 5. Conclusions

Aquest projecte ens ha servit per assimilar els coneixements teòrics que havíem après anteriorment però que fins ara, no hem tingut l'oportunitat de posar-los a prova.

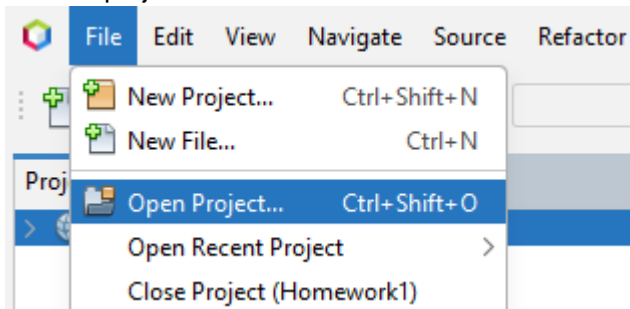
Em aconseguit assolir tots els objectius que ens havíem plantejat, al principi es pot dificultar el fet entendre com funciona la gestió d'una API web, però finalment es veu bastant pràctic i útil per en un futur augmentar la dificultat i poder gestionar una base de dades més complexa.

Un dels aspectes més importants que hem après és com integrar una base de dades a l'API per emmagatzemar i recuperar dades de manera dinàmica. La gestió de l'autenticació i autorització dels usuaris també ha estat un repte que ens ha permès conèixer millor els mètodes de seguretat en les aplicacions web.

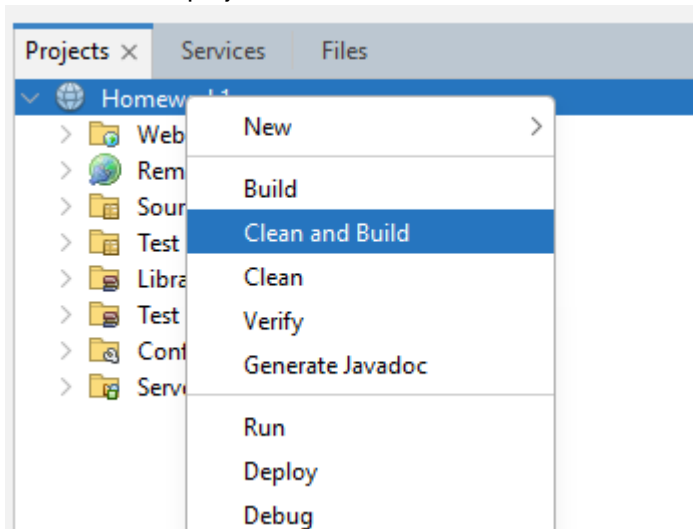
Amb el desenvolupament d'aquesta pràctica ens hem adonat l'importància de fer un joc de proves eshaustiu, ja que és essencial per garantir el correcte funcionament de tots els mètodes que hem creat, i així que l'usuari pugui fer un correcte ús d'aquesta API. A partir d'aquest projecte base podem desenvolupar un bon front-end.

## 6. Manual d'instal·lació

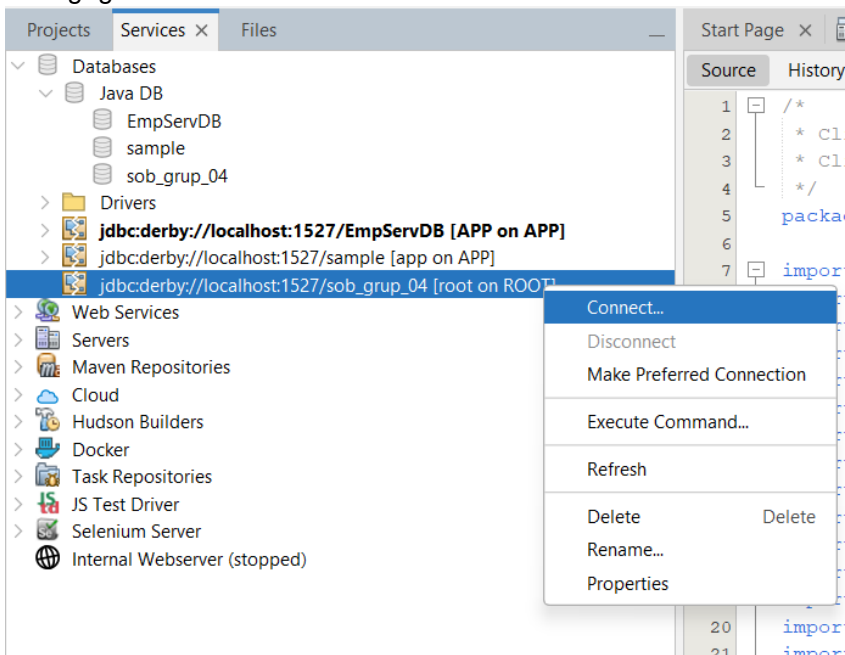
### 1. Obrir projecte al NetBeans



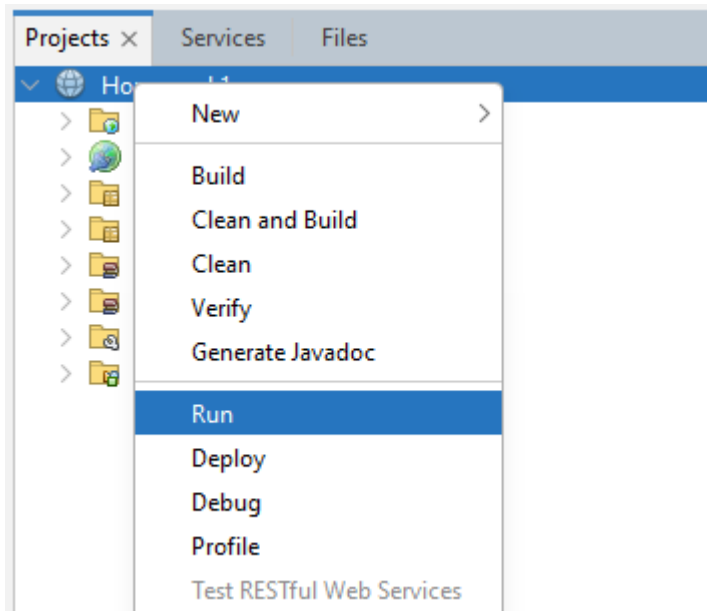
### 2. Click dret al projecte i donar-li a "Clean and build"



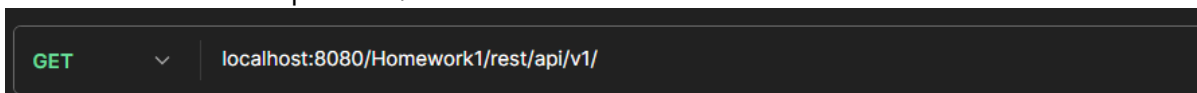
### 3. Engegar el servidor



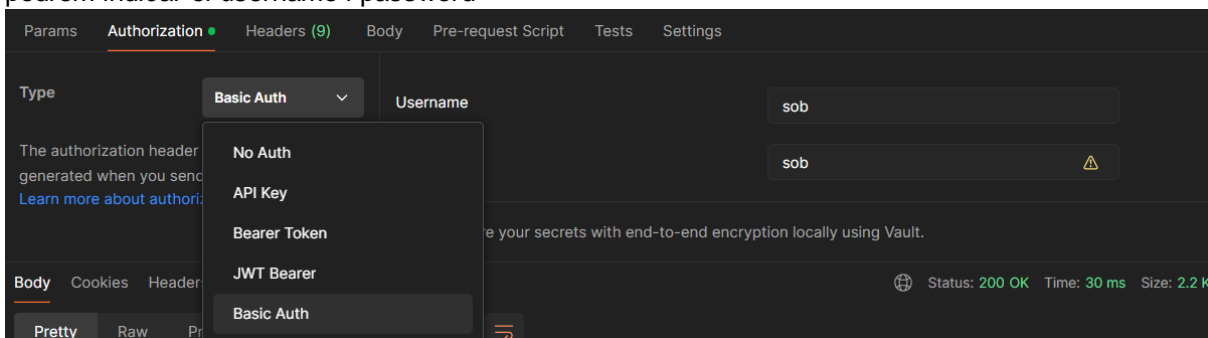
### 4. Click dret al projecte i donar-li a "Run"



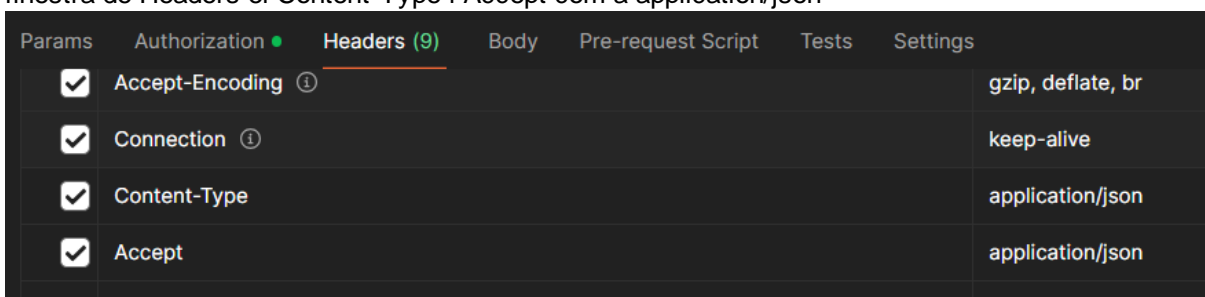
5. Obrir Postman on podem realitzar el joc de porves, indicant la següent URL per fer les peticions a la API REST. Si volem actuar sobre els articles haurem de especificar /article i si volem actual sobre els usuaris haurem d'especificar /customer



6. Per autenticar-se, haurem d'anar a la finestra de Authorization i indicar BasicAuth com a Type i podrem indicar el username i password



7. Per treballar sobre JSON i poder rebre les respostes en aquest format haurem d'indicar en la finestra de Headers el Content-Type i Accept com a application/json



8. Mostrem un GET dels articles com a exemple, si volem un article en qüestió només haurem de especificar al final de la URL /{id} , indicant el id del article

localhost:8080/Homework1/rest/api/v1/article

GET localhost:8080/Homework1/rest/api/v1/article Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (4) Test Results Status: 200 OK Time: 168 ms Size: 2.2 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "dataPublicacio": "01-12-2024",
4     "privat": false,
5     "titol": "Titol del article 7",
6     "id": 7,
7     "temes": [
8       {
9         "id": 1,
10        "nom": "Computer Science"
11      },
12      {
13        "id": 2,
14        "nom": "Informatica"
15      }
16    ],
17    "visualitzacions": 0,
18    "autor": {
19      "nom": "Carlos",
```

9. Tant per fer un POST com un PUT ens haurem d'anar a la finestra superior Body i marcarem 'raw'.

POST localhost:8080/Homework1/rest/api/v1/article Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   {
3     "dataPublicacio": "01-12-2024",
4     "privat": false,
5     "titol": "Titol exemple",
6     "temes": [
7       {
8         "id": 1,
9         "nom": "Computer Science"
10      },
11      {
12        "id": 2,
13        "nom": "Informatica"
14      }
15    ],
16    "resum": "Resum exemple",
17    "imatge": "imatge exemple"
18  }
19 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 168 ms Size: 2.2 KB Save Response