# Definició NodeJS API projecte ImagIA II

## Índex

# Comú a totes crides a l'API

| Common to all API endpoints | | |
|---|---|---|
| OUTPUT params | Description | Examples |
| status | Success or error in the API call. Valid values are OK, ERROR | "OK"<br>"ERROR" |
| message | String with details about completion or error in the execution of the call. | "PIN is required"<br>"session_token expired" |
| data | String representing a JSON object with any additional data that needs to be sent as a response | {<br>"email": "user@ex.com"<br>} |

**api_key*** : cal enviar-la a través de la capçalera en totes les crides en les que sigui requerida:
Authorization: Bearer ABCD1234EFGH5678IJKL

# API Definició d'endpoints

Registre de l'usuari en el sistema

| Endpoint: **POST /api/usuaris/registrar** | | |
|---|---|---|
| INPUT params | Description | Examples |
| telefon | String with the user's phone number | "+34 600 000 000" |
| nickname | String with the nickname by which the user will be identified in the system | "SparkleFuzzMcGee" |
| email | String with the user's email address | "user@example.com" |
| | | |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "L'usuari s'ha creat correctament" |
| data | Object representing the new user that has been created | |

Example call:

```
$ curl -X POST http://example.com/api/usuaris/registrar \
-H "Content-Type: application/json" \
-d '{
  "telefon": "+34 600 000 000",
  "nickname": "SparkleFuzzMcGee",
  "email": "user@example.com"
}'
```

```
{"status": "OK", "message": "L'usuari s'ha creat correctament", "data": {"nickname": "SparkleFuzzMcGee", "email": "user@example.com"}}
```

Validació d'usuari: Un cop l'usuari s'intenta registar, rep un missatge SMS amb un número que cal validar a través d'aquesta crida.

| Endpoint: **POST /api/usuaris/validar** | | |
|---|---|---|
| INPUT params | Description | Examples |
| telefon | String with the user's phone number | "+34 600 000 000" |
| codi_validacio | Integer with 6 digits representing the validation code that the user must provide to validate their account ( Sent by the Android application and forwarded by the NodeJS Server) | 817921 |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "L'usuari s'ha validat correctament" |
| data | Object containing the response data. For this operation, it includes a generated api_key. | {"status": "OK", "message": "Usuari validat correctament", "data": {"api_key": "ABCD1234EFGH5678IJKL"}} |

Example call:

```
$ curl -X POST http://example.com/api/usuaris/validar \
-H "Content-Type: application/json" \
-d '{
  "telefon": "+34 600 000 000",
  "codi_validacio": 817921
}'
```

```
{"status": "OK", "message": "Usuari validat correctament", "data": {"api_key": "ABCD1234EFGH5678IJKL"}}
```

Obtenció de la informació de l'usuari que correspon a l'API key

| Endpoint: **GET /api/usuaris/perfil** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer YOUR_API_KEY_HERE | | |
| INPUT params | Description | Examples |
| (no aplica) | | |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Informació d'usuari obtinguda correctament" |
| data | Object containing the response data. For this operation, it includes the data corresponding to the user linked to the API key | (See the example call) |

Example call:

```
$ curl -X GET http://example.com/api/usuaris/perfil \
-H "Authorization: Bearer ABCD1234EFGH5678IJKL"
```

```
{
  "status": "OK",
  "message": "Informació de l'usuari obtinguda correctament",
  "data": {
    "nickname": "SparkleFuzzMcGee",
    "email": "user@example.com",
    "telefon": "+34 600 000 000",
    "validat": true,
    "tos": true,
    "pla": "premium"
    // Opció per ampliar amb altres camps rellevants sobre l'usuari en el futur.
  }
}
```

**Nota sobre Seguretat:** En un entorn real seria crucial assegurar-se que la comunicació entre el client i el servidor sigui segura. Això normalment s'aconsegueix mitjançant l'ús d'HTTPS, que encripta la informació enviada a través de la xarxa, inclosa la clau d'API en la capçalera d'Authorization.

Obté la quota disponible de l'usuari que realitza la petició (API KEY). Pensada per comprovar si l'usuari la pot realitzar. Error 429 si l'usuari ja ha esgotat la quota diaria.

| Endpoint: **GET /api/usuaris/quota** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer YOUR_API_KEY_HERE | | |
| INPUT params | Description | Examples |
| (no params) | | |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Quota consultada correctament" |
| data | Object containing the response data. For this operation, it includes the data corresponding to the user linked to the API key | (See the example call) |

Example call:

```
$ curl -X GET http://example.com/api/usuaris/quota \
-H "Authorization: Bearer ABCD1234EFGH5678IJKL"
```

```
{
  "status": "OK",
  "message": "Quota consultada correctament",
  "data": {
    "pla": "premium",
    "quota": {
        "total": 20,
        "consumida": 15,
        "disponible": 5
    }
  }
}
```

Canvia el pla de l'usuari i les quotes d'acord amb el nou pla. Disponible només per usuaris Administradors que s'hagin autenticat i facilitin la seva API_KEY. No cal facilitar tots els camps de l'usuari indicats com a opcionals, només un que permeti identificar l'usuari.

| Endpoint: **POST /api/admin/usuaris/pla/actualitzar** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer `YOUR_ADMIN_API_KEY_HERE` | | |
| INPUT params | Description | Examples |
| telefon (optional) | String with the user's phone number | "+34 600 000 000" |
| nickname (optional) | String with the nickname by which the user will be identified in the system | "SparkleFuzzMcGee" |
| email (optional) | String with the user's email address | "user@example.com" |
| pla | String that identifies the new user plan | "premium" |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Pla canviat correctament" |
| data | Object containing the response data. For this operation, it includes the data corresponding the new plan and quotas applicable to the user. | (See the example call) |

Example call:

```
$ curl -X POST http://example.com/api/admin/usuaris/pla/actualitzar \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_ADMIN_API_KEY_HERE" \
-d '{
  "nickname": "SparkleFuzzMcGee", "pla": "premium"
}'
```

```
{ "status": "OK",
  "message": "Pla canviat correctament",
  "data": {
    "pla": "premium",
    "quota": { "total": 20, "consumida": 15, "disponible": 5 }
  }
}
```

Consulta la quota d'un usuari. Disponible només per usuaris Administradors que s'hagin autenticat i facilitin la seva API_KEY. No cal facilitar tots els camps de l'usuari indicats com a opcionals, només un que permeti identificar l'usuari.

| Endpoint: **GET /api/admin/usuaris/quota** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer `YOUR_ADMIN_API_KEY_HERE` | | |
| INPUT params | Description | Examples |
| telefon (optional) | String with the user's phone number | "+34 600 000 000" |
| nickname (optional) | String with the nickname by which the user will be identified in the system | "SparkleFuzzMcGee" |
| email (optional) | String with the user's email address | "user@example.com" |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Quota d'usuari obtinguda correctament" |
| data | Object containing the response data. For this operation, it includes the data corresponding to the plan and quotas applicable to the user. | (See the example call) |

Example call:

```
$ curl -X POST http://example.com/api/admin/usuaris/quota \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_ADMIN_API_KEY_HERE" \
-d '{
  "nickname": "SparkleFuzzMcGee"
}'
```

```
{
  "status": "OK",
  "message": "Quota d'usuari obtinguda correctament",
  "data": {
    "pla": "premium",
    "quota": { "total": 20, "consumida": 15, "disponible": 5 }
  }
}
```

Estableix valors de quota per un usuari. Disponible només per usuaris Administradors que s'hagin autenticat i facilitin la seva API_KEY. No cal facilitar tots els camps de l'usuari indicats com a opcionals, només un que permeti identificar l'usuari.

| Endpoint: **POST /api/admin/usuaris/quota/actualitzar** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer `YOUR_ADMIN_API_KEY_HERE` | | |
| INPUT params | Description | Examples |
| telefon (optional) | String with the user's phone number | "+34 600 000 000" |
| nickname (optional) | String with the nickname by which the user will be identified in the system | "SparkleFuzzMcGee" |
| email (optional) | String with the user's email address | "user@example.com" |
| limit (optional) | The total daily quota | 30 |
| disponible (optional) | The quota available for the current day. | 20 |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Quota d'usuari actualitzada corretament" |
| data | Object containing the response data. For this operation, it includes the data corresponding to the plan and quotas applicable to the user. | (See the example call) |

Example call:

```
$ curl -X POST http://example.com /api/admin/usuaris/quota/actualitzar \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_ADMIN_API_KEY_HERE" \
-d '{ "nickname": "SparkleFuzzMcGee", "limit": 30, "disponible": 20 }'
```

```
{ "status": "OK",
  "message": "Quota d'usuari actualizada correctament",
  "data": { "pla": "premium",
            "quota": { "total": 20, "consumida": 15, "disponible": 5 }
  }
}
```

Obté la llista completa dels usuaris. Disponible només per usuaris Administradors que s'hagin autenticat i facilitin la seva API_KEY.  Retorna la llista completa d'usuaris i els grups als que pertanyen.

| Endpoint: **GET /api/admin/usuaris** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer `YOUR_ADMIN_API_KEY_HERE` | | |
| INPUT params | Description | Examples |
| (no params) | | |
| OUTPUT params | Description | Examples |
| status | (see above) | "OK" , "ERROR" |
| message | (see above) | "Consulta realitzada correctament" |
| data | Object containing the response data. For this operation, it includes the full list of users | (See the example call) |

Example call:

```
$ curl -X GET http://example.com/api/admin/usuaris \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_ADMIN_API_KEY_HERE" \
-d '{ "nickname": "SparkleFuzzMcGee", "limit": 30, "disponible": 20 }'
```

```
{ "status": "OK",
  "message": "Consulta realitzada correctament",
  "data": [
    {
      "nickname": "SparkleFuzzMcGee", "email": "user1@example.com",
      "telefon": "+34 600 000 001", "validat": true,
      "tos": true, "pla": "premium",
      "grups": ["Desenvolupadors", "Beta Testers"],
      "quota": {
        "total": 100,
        "consumida": 20,
        "disponible": 80
      }
    },
    (More data…)
}
```

Quan un usuari administrador facilita les credencials correctes i es valida en el sistema se li retorna una api_key amb la que podrà realitzar crides a endpoints destinats als administradors.

| Endpoint: **POST /api/admin/usuaris/login** | | |
|---|---|---|
| INPUT params | Description | Examples |
| email | User's email address | usuari@example.com |
| contrasenya | User's password | No “123456” |
| OUTPUT params | Description | Examples |
| status | (see above) | “OK” , “ERROR” |
| message | (see above) | “L’usuari s’ha autenticat correctament” |
| data | Object containing the response data. For this operation, it includes a generated api_key. | {"status": "OK", "message": "Usuari autenticat correctament", "data": {"api_key": "D23qswfSgR6VM9cuTuN" }} |

Example call:

```
$ curl -X POST http://example.com/api/admin/usuaris/login \
-H "Content-Type: application/json" \
-d '{
  "email": "user@example.com",
  "password": "Not '123456'"
}'
```

```
{"status": "OK", "message": "Usuari autenticat correctament", "data": {"api_key": "D23qswfSgR6VM9cuTuN"}}
```

**Nota important:** Aquesta functionalitat d’autenticació d’usuaris es una versió simplificada de la que seria recomanable en aplicacions professionals. Veure (per curiositat, no per implementar) Recomanacions login d’usuaris administradors

Analitzar i descriure el contingut d'una imatge. Només admet una única imatge.

| Endpoint: **POST /api/analitzar-imatge** | | |
|---|---|---|
| HEADERS | | |
| Authorization: Bearer YOUR_API_KEY_HERE | | |
| INPUT params | Description | Examples |
| prompt | Text that asks for an action about the image | Describe this image |
| Octet-stream files | List of images | [] Tot i que sigui una llista el model llama3.2 vision només admet una imatge |
| stream | boolean | |
| OUTPUT params | Description | Examples |
| status | | OK, ERROR |
| message | | Maria image processed |
| data | | |

Example call:

```
curl -X POST "https://api.example.com/api/analitzar-imatge" \
  -H "Authorization: Bearer YOUR_API_KEY_HERE" \
  -H "Content-Type: application/json" \
  -d '{
    "prompt": "Què hi ha en aquesta imatge?",
    "images": [
      "base64_encoded_image_1",
      "base64_encoded_image_2"
    ],
    "stream": false,
    "model": "llama3.2-vision:latest"
  }'
```

```
{
  "status": "OK",
  "message": "Imatges processades correctament",
  "data": {
    "token": "AD598745GAS25ZR",
    "description": "La imatge mostra...",
    "processing_time": "2.3s",
    "model_used": "llama3.2-vision:latest"
  }
}
```

```
{
  "status": "ERROR",
  "message": "Token invàlid",
  "data": null
}
```

# Annexos

## Authorization: Bearer amb NodeJS (Client)

Des d'un servidor NodeJS, per realitzar en la que s'indica la API_KEY com la d'aquest exemple

```
curl -X POST http://example.com/api/peticions/afegir \
-H "Content-Type: application/json" \
-H "Authorization: Bearer ABCD1234EFGH5678IJKL" \
-d '{
  "model": "llama3.2-vision:latest",
  "prompt": "What is in this picture?",
  "imatges": ["base64_encoded_image1"]
}'
```

Podem usar el mòdul **"axios"** que és molt popular per fer crides HTTP des de Node.js. Aquí hi ha un exemple sobre com fer la crida post a l'endpoint

```
const axios = require('axios');

// Funció per provar la petició
async function provarPeticioImatge() {
  try {
    const peticioData = {
      model: "llama3.2-vision:latest",
      prompt: "What is in this picture?",
      imatges: ["base64_encoded_image1"] // Aquí aniria la imatge real en base64
    };

    const response = await axios.post('http://localhost:3000/api/peticions/afegir',
      peticioData,
      {
        headers: {
          'Content-Type': 'application/json',
          'Authorization': 'Bearer ABCD1234EFGH5678IJKL'
        }
      }
    );

    console.log('Resposta exitosa:', response.data);
  } catch (error) {
    if (error.response) {
      // L'error ve del servidor amb un status diferent de 2xx
      console.error('Error de resposta:', error.response.data);
```

```
    } else if (error.request) {
      // La petició es va fer però no es va rebre resposta
      console.error('Error de petició:', error.request);
    } else {
      // Error en la configuració de la petició
      console.error('Error:', error.message);
    }
  }
}

// Prova amb token invàlid
async function provarPeticioInvalida() {
  try {
    const peticioData = {
      model: "llama3.2-vision:latest",
      prompt: "What is in this picture?",
      imatges: ["base64_encoded_image1"]
    };

    const response = await axios.post('http://localhost:3000/api/peticions/afegir',
      peticioData,
      {
        headers: {
          'Content-Type': 'application/json',
          'Authorization': 'Bearer TOKEN_INVALID'
        }
      }
    );
  } catch (error) {
    console.log('Resposta d\'error esperada:', error.response?.data);
  }
}

// Executar les proves
provarPeticioImatge();
provarPeticioInvalida();
```

# Authorization: Bearer amb NodeJS (Server)

Implementació d'un servidor Express.js que gestiona peticions d'imatges amb IA mitjançant un endpoint protegit. Inclou autenticació amb token Bearer, validació dels paràmetres de la petició (model, prompt i imatges) i retorna respostes en un format estandarditzat.

```
const express = require('express');
const app = express();

app.use(express.json({ limit: '50mb' })); // Augmentem el límit per les imatges en base64

// Funció auxiliar per generar un format de resposta estàndard
const createResponse = (status, message, data = null) => {
  return {
    status: status,
    message: message,
    data: data
  };
};

// Middleware per validar el contingut de la petició
const validateRequestContent = (req, res, next) => {
  const { model, prompt, imatges } = req.body;

  // Validació dels camps requerits
  if (!model) {
    return res.status(400).json(
      createResponse('ERROR', 'El camp model és obligatori')
    );
  }

  if (!prompt) {
    return res.status(400).json(
      createResponse('ERROR', 'El camp prompt és obligatori')
    );
  }

  if (!imatges || !Array.isArray(imatges) || imatges.length === 0) {
    return res.status(400).json(
      createResponse('ERROR', 'Es requereix almenys una imatge')
    );
  }

  // Validació del model específic
  if (model !== 'llama3.2-vision:latest') {
    return res.status(400).json(
      createResponse('ERROR', 'Model no suportat')
    );
  }
```

```
  next();
};

// Middleware per l'autenticació amb token Bearer
const authMiddleware = (req, res, next) => {
  const authHeader = req.headers['authorization'];

  if (!authHeader) {
    return res.status(401).json(
      createResponse('ERROR', 'Es requereix la capçalera d\'autorització')
    );
  }

  if (!authHeader.startsWith('Bearer ')) {
    return res.status(401).json(
      createResponse('ERROR', 'Format d\'autorització invàlid')
    );
  }

  const token = authHeader.substring(7);

  if (token !== 'ABCD1234EFGH5678IJKL') {
    return res.status(401).json(
      createResponse('ERROR', 'Token invàlid')
    );
  }

  // Si tot és correcte, continua
  next();
};

// Ruta per processar les peticions d'imatges
app.post('/api/peticions/afegir', [authMiddleware, validateRequestContent], async (req, res) => {
  try {
    const { model, prompt, imatges } = req.body;

    // Aquí aniria la lògica per processar les imatges amb el model
    // Per aquest exemple, simplement retornem una resposta exitosa

    res.json(
      createResponse('OK', 'Petició processada correctament', {
        descripció: '.....'
      })
    );
  } catch (error) {
    console.error('Error processant la petició:', error);
    res.status(500).json(
      createResponse('ERROR', 'Error intern processant la petició')
```

```
    );
  }
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Servidor executant-se al port ${PORT}`);
});
```

# Exemple de resposta Ollama i llama 3.2-vision (stream true o false)

A continuació es mostren dos exemples de respostes d'Ollama amb l'opció streaming activada (envia paraules a mesura que les genera) o desactivada-

## Sense stream

{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:50:14.220787031Z","response":"La imatge mostra un personatge pixelat, probablement del joc Super Mario Bros., que es caracteritza per les seves sabates amb cordills i els seus pantalons vermells. El color de la barba també és distintiu.\n\nL'estil pixelat i el disseny de l'imatge fan pensar en un personatge dels jocs Super Mario Bros., probablement Mario o Luigi. No obstant això, no hi ha prou detalls per identificar amb certesa qui és el personatge.","done":true,"done_reason":"stop","context":[128006,882,128007,271,58,1931,12,15,60,128256,271,29401,30588,658,1744,15960,6520,264,1208,737,266,713,128009,128006,78191,128007,271,8921,737,266,713,93197,653,1732,266,713,13252,266,11,35977,479,1624,503,511,7445,24270,34321,2637,1744,1560,33329,275,4458,824,3625,513,2396,19972,988,9049,23125,3385,602,21371,42498,26346,278,2439,96998,6572,13,4072,1933,409,1208,3703,4749,17834,978,22257,74478,19260,382,43,17771,321,13252,266,602,658,834,12021,88,409,326,6,78432,713,8571,94151,665,653,1732,266,713,1624,82,503,14460,7445,24270,34321,2637,35977,479,24270,297,83183,13,2360,1536,4811,264,953,22980,11,912,15960,6520,550,283,3474,5700,824,3608,24123,9049,2847,23174,7930,22257,658,1732,266,713,13],"total_duration":10430143054,"load_duration":1911214009,"prompt_eval_count":24,"prompt_eval_duration":3883000000,"eval_count":120,"eval_duration":4546000000}

## Amb stream

{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:01.442648723Z","response":"L","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:01.480854304Z","response":"'","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:01.519203441Z","response":"imat","done":false}
…
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.585357455Z","response":" pel","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.624167972Z","response":"·","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.663062254Z","response":"l","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.701928789Z","response":"íc","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.740752319Z","response":"ules","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.779582329Z","response":".","done":false}
{"model":"llama3.2-vision:latest","created_at":"2025-01-21T08:53:28.818490459Z","response":"","done":true,"done_reason":"stop","context":[128006,882,128007,271,58,1931,12,15,60,128256,271,29401,30588,658,1744,15960,6520,264,1208,737,266,713,128009,128006,78191,128007,271,43,6,78432,713,93197,24270,11,653,1732,266,713,409,2835,73,14460,61344,337,455,1900,824,23643,13,29124,82,653,1624,82,46684,288,296,5512,390,797,6256,602,5526,82,409,2458,82,21371,27138,665,658,108130,1624,82,2835,73,14460,13,5034,513,6723,58316,3457,1832,11412,5899,404,326,1092,264,1208,44769,79518,409,7445,24270,34321,2637,9507,276,3209,2649,658,220,3753,20,824,74313,13,445,6,4775,268,294,26248,724,1732,266,713,1560,1323,43150,66079,453,220,3753,16,11,100809,1443,7420,84,67559,57960,11,653,29520,46602,292,61344,337,455,5477,409,23643,11,11412,834,12021,53857,653,503,511,459,6431,266,4418,798,18711,13,2998,264,724,2835,73,511,11,658,34086,9265,11639,653,2162,1744,31081,689,1826,266,10773,519,14098,288,294,22827,46298,321,14260,4355,13,67559,57960,11412,48837,264,724,1732,266,713,824,447,4558,912,5899,689,2107,94660,1744,18728,11,602,11412,293,349,19571,27578,9049,658,9859,409,29888,1543,11,12203,1744,665,15715,6572,14269,21371,23200,67987,7661,3675,12290,277,824,3625,63886,630,288,13,362,31632,409,9507,51585,11,658,20607,9859,649,10176,6496,264,24270,382,79834,22257,653,2162,28705,275,294,6,8733,6730,10994,10707,6885,11,7120,26346,278,2439,96998,6572,602,7120,5568,73,2439,1099,70404,13,350,978,653,66803,72,96998,616,29832,602,62090,5203,46298,969,10707,6885,9049,653,28800,409,743,2483,64347,665,76,343,13,362,296,5512,11,47289,7120,53396,82,86696,602,21236,2439,1744,908,1541,268,5203,2847,64,3237,344,48877,453,20607,938,67523,

13,445,6,62078,533,19260,9302,294,26248,724,1732,266,713,22257,1208,513,6723,4443,48877,11,12203,1744,6520,1826,266,91439,4186,11289,2649,824,264,4097,277,27578,665,21371,2835,73,14460,382,79834,22257,658,34086,9265,409,2458,82,21371,503,14460,409,7445,24270,34321,2637,602,17834,978,56270,953,264,4902,417,274,4558,4108,470,83183,596,83396,11,2999,13,24270,297,24270,61197,13,362,961,294,26248,724,288,274,4558,4108,11,17834,978,6520,56270,70,332,665,4902,417,2835,73,14460,409,23643,1744,912,1826,276,47105,1900,9049,1208,274,4558,7379,409,24270,382,32,296,5512,11,658,1732,266,713,409,24270,6520,1826,266,4186,11289,266,824,264,2773,511,290,277,2027,288,1624,34670,23643,13,2998,21371,904,82,220,1954,11,11412,9507,276,3209,277,5203,32850,21557,409,31225,84,1441,602,4902,417,1665,288,1744,5899,3675,470,264,2385,658,20607,834,12021,88,13,4072,220,1049,20,11,1208,274,4558,7379,409,24270,1560,11412,5625,404,665,653,47015,824,10942,2835,73,14460,409,40292,13482,11,389,7661,689,94360,7962,9049,658,2138,29832,1732,266,713,382,6719,1732,266,713,6520,1826,266,4186,11289,266,665,4902,417,3869,3172,1220,470,12077,14260,75,30515,2482,11,459,1769,297,29876,76,1233,13,4072,220,2550,18,11,1208,56092,91065,10126,86285,3233,64,88153,6007,11412,9507,276,3209,277,5203,274,4558,7379,409,42168,84,953,437,4039,1900,3122,2649,665,24270,602,21371,42498,1097,1233,13,362,724,288,274,4558,4108,19538,1826,266,91439,5526,82,264,2458,658,108130,382,1737,84082,35994,11,24270,22257,653,1732,266,713,1744,6520,43327,332,653,16109,11676,20854,453,9507,867,1624,82,904,82,13,14433,56270,70,332,665,22337,2641,2835,73,14460,409,23643,602,6520,1826,266,4186,11289,266,824,264,2773,511,290,277,2027,288,1624,34670,13,362,296,5512,11,6520,43327,332,5203,16109,1685,4558,19489,25098,6496,83055,9049,274,4558,4108,409,42168,84,953,437,4039,1900,297,12077,14260,75,30515,2482,13],"total_duration":27504131565,"load_duration":14924303,"prompt_eval_count":24,"prompt_eval_duration":51000000,"eval_count":707,"eval_duration":27376000000}

# Exemple amb Java tractament resposta en stream

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Base64;
import org.json.JSONArray;
import org.json.JSONObject;

public class OllamaClient {
  public static void main(String[] args) {
     // Ruta a la imatge (equivalent al PowerShell)
     String imagePath = "../../data/exemples/mario_128px.jpg";

     try {
        // Verificar que l'arxiu existeix
        Path path = Paths.get(imagePath);
        if (!Files.exists(path)) {
           System.err.println("La imatge no existeix a la ruta especificada");
           return;
        }

        // Llegir la imatge i convertir-la a base64
        byte[] imageBytes = Files.readAllBytes(path);
        String imageBase64 = Base64.getEncoder().encodeToString(imageBytes);

        // Crear l'objecte JSON (equivalent al PowerShell $jsonBody)
        JSONObject jsonBody = new JSONObject();
        jsonBody.put("model", "llama3.2-vision:latest");
        jsonBody.put("prompt", "Identifica el que hi ha a la imatge");
        jsonBody.put("images", new JSONArray().put(imageBase64));
        jsonBody.put("stream", true);

        // Configurar la connexió HTTP
        URL url = new URL("http://127.0.0.1:11434/api/generate");
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setDoOutput(true);

        // Enviar la petició
        try (OutputStream os = conn.getOutputStream()) {
           byte[] input = jsonBody.toString().getBytes("UTF-8");
           os.write(input, 0, input.length);
        }

        // Llegir la resposta en streaming
        System.out.println("Enviant petició a Ollama...");
        try (BufferedReader br = new BufferedReader(
              new InputStreamReader(conn.getInputStream(), "UTF-8"))) {

           String line;
```

```
            while ((line = br.readLine()) != null) {
                try {
                    JSONObject response = new JSONObject(line);
                    if (response.has("response")) {
                        System.out.print(response.getString("response"));
                    }
                } catch (Exception e) {
                    System.err.println("Error processant la resposta: " + line);
                }
            }
        }

        conn.disconnect();

    } catch (Exception e) {
        System.err.println("Error en processar la imatge: " + e.getMessage());
        e.printStackTrace();
    }
  }
}
```

# Recomanacions login d'usuaris administradors

Útil per la vostra vida professional en general

En un entorn professional, la forma recomanada perquè un usuari administrador s'autentiqui en el sistema a través d'una API i posteriorment realitzi peticions validades a diferents endpoints reservats per a administradors involucra diversos passos i pràctiques de seguretat. Aquí tens una guia pas a pas:

1. **Autenticació Segura**
    - **Credencials Segures:** Assegura't que les credencials (com ara contrasenyes) siguin fortes i emmagatzemades de manera segura utilitzant hash i sal. No emmagatzemar les contrasenyes en text pla.
    - **Protocol HTTPS:** Utilitza sempre HTTPS per a totes les comunicacions entre el client i el servidor per garantir que les dades estiguin xifrades durant el trànsit.
2. **Emisió d'un Token d'Accés**
    - **JWT (JSON Web Tokens):** Una pràctica comuna és utilitzar JWTs per a l'emissió de tokens d'accés. Quan un usuari administrador s'autentica amb èxit, el sistema genera un JWT com a api_key. Aquest token pot incloure reclamacions (claims) com ara l'identificador de l'usuari, el rol (administrador) i la caducitat del token.
    - **Caducitat:** Els tokens d'accés haurien de tenir un període de caducitat curt per minimitzar el risc en cas que es comprometin.
3. **Validació i Autorització per a Cada Petició**
    - **Middleware d'Autorització:** Implementa un middleware a l'API que validi el token d'accés en cada petició a endpoints restringits. Aquest middleware hauria de verificar la signatura del token, la seva caducitat i si l'usuari té permisos per accedir a l'endpoint sol·licitat.
    - **Gestió de Sessions:** Manté un registre de sessions actives i ofereix mecanismes per a la invalidació de tokens, permetent als usuaris tancar la sessió activament o invalidar tokens en cas de sospita de compromís.
4. **Control d'Accés Basat en Rols**
    - **RBAC (Role-Based Access Control):** Utilitza un sistema de control d'accés basat en rols per definir quins endpoints estan disponibles per a quins rols. Això permet una gestió fina dels permisos d'accés per a diferents tipus d'usuaris dins del sistema.

5. **Registre i Monitoratge**

    - **Auditoria:** Manté un registre detallat de totes les peticions d'accés, incloent les autenticacions reeixides i fallides, així com les peticions als endpoints. Això facilita la detecció d'activitats sospitoses i l'auditoria de seguretat.

6. **Actualització i Manteniment**

    - Seguretat Contínua: Manté les llibreries i dependències actualitzades a les seves últimes versions per protegir-te contra vulnerabilitats conegudes. Implementa revisions de seguretat regulars i proves de penetració per identificar i solucionar possibles debilitats.

# Codis HTTP freqüents en API

Útil per la vostra vida professional en general

- **200 OK:** La petició ha tingut èxit. Aquest és el codi d'estat més comú retornat quan tot ha anat bé.
- **201 Created:** La petició ha tingut èxit i s'ha creat un nou recurs com a resultat. Això és comú en operacions POST que resulten en la creació d'un nou recurs.
- **202 Accepted:** La petició ha estat acceptada per ser processada, però el processament encara no ha finalitzat.
- **204 No Content:** La petició ha tingut èxit, però el servidor no retorna cap contingut. Això és comú en operacions DELETE.
- **400 Bad Request:** El servidor no pot o no processarà la petició degut a alguna cosa que es perceb com a error del client (p.ex., sintaxi malformada, mida massa gran, dades no vàlides).
- **401 Unauthorized:** Aquest estat indica que l'autenticació és necessària i ha fallat o encara no ha estat proporcionada.
- **402 Payment Required:** Reservat per a ús futur. La intenció original era per a casos on el contingut d'accés requeria pagament, però no s'ha utilitzat àmpliament en aquest context.
- **403 Forbidden:** El servidor està rebutjant respondre a la petició. A diferència del 401, l'identificació del client no ajudarà i la sol·licitud s'hauria de no repetir.
- **404 Not Found:** El servidor no ha trobat res que correspongui a l'URI de la petició. Això significa que la URL sol·licitada no existeix en el servidor.
- **405 Method Not Allowed:** El mètode especificat en la petició no és permès per al recurs identificat per la URL.
- **429 Too Many Requests:** L'usuari ha enviat massa peticions en un període de temps donat. Aquest estat s'utilitza per controlar la limitació de taxa en els servidors per prevenir l'abús de les API.
- **500 Internal Server Error:** Un missatge genèric, donat quan no hi ha cap missatge més específic adequat. Indica errors inesperats del servidor.
- **501 Not Implemented:** El servidor no suporta una funcionalitat requerida per complir la petició.
- **502 Bad Gateway:** El servidor, mentre actua com a porta d'enllaç o proxy, va rebre una resposta invàlida des d'un servidor upstream al qual va accedir en intentar complir la petició.
- **503 Service Unavailable:** El servidor no està disponible actualment (sobrecarregat o avall per manteniment). Generalment, això és temporal.