



# UNIVERSITÉ DE LORRAINE

## Documentation fonctionnelle Groupe 2 : Surveys

---

### Membres :

*CONRAD Alexandre*

*DAVIES Liam*

*CHEVALET Clément*

*BIR Vincent*

---

## Table des matières

Membres : .....	1
Présentation du logiciel .....	4
Prérequis .....	4
Installation .....	4
Accessibilité.....	5
Documentations.....	6
Comment utiliser notre projet.....	7
Partie GET :.....	8
Partie POST :.....	9
Partie PUT : .....	10
Partie PATCH : .....	12
Partie DELETE : .....	13
Postman .....	14
Prérequis Postman .....	14
Importation du fichier Postman.....	14
Interface de Postman.....	15
Faire des requêtes à notre API.....	17
Date avec le plus de personnes présentes.....	17
Date avec le plus de personnes potentiellement présentes .....	17
Supprimer un choix .....	18
Afficher la liste des choix possibles.....	18
Ajouter un choix pour un sondage.....	19
Récupérer tous les commentaires d'un sondage .....	19
Ajouter un commentaire sur un sondage .....	20
Récupérer la liste de tous les sondages .....	20
Créer un nouveau sondage .....	21
Récupérer la liste des sondages actifs .....	22
Récupérer la liste des sondages inactifs .....	22
Supprimer un sondage .....	23
Récupérer un sondage spécifique.....	23
Clôturer un sondage avant la date de fin.....	24
Modifier les informations d'un sondage.....	24
Récupérer toutes les options .....	25
Ajout d'une option .....	26
Récupérer tous les votes d'un choix .....	26

Récupérer tous les votes d'une option .....	27
Participation d'un utilisateur .....	28

## Présentation du logiciel

Le logiciel est un projet réalisé par les membres du groupe 2 du Master 2 GI Informatique 2020-2021 encadré par Mr. Cédric Moschetta. L'objectif de ce logiciel est de réaliser une API pour des sondages en ligne. Le programme doit respecter certaines spécificités techniques comme :

- Un tunnel d'intégration
- Utilisation de Git
- Utilisation de Java 11
- Maven
- Spring
- Et différentes technologies étudiées en cours (H2, Hibernate, Lombok, Swagger)

L'application permet une gestion complète des sondages, des votes, des choix, des commentaires et de l'analyse des réponses (date qui a le meilleur potentiel).

## Prérequis

Afin d'utiliser notre logiciel, il suffit d'avoir la liste des applications ou logiciels ci-dessous :

- Java 11
- Le lien de notre projet Git : <https://gitlab.cedricmtta.com/Alexandre/surveys/-/tree/master/>
- Un éditeur de code

## Installation

Dans un premier lieu, l'installation du projet se fait en clonant notre projet Git.

Dans un terminal de commande windows, il suffit de taper la ligne de code suivante :

```
git clone ssh://git@gitlab.cedricmtta.com:5022/Alexandre/surveys.git
```

Il vous suffira d'ouvrir le dossier du projet téléchargé à l'aide d'un éditeur de code (IntelliJ par exemple), et de le lancer via l'une des configurations installées par défaut.

## Accessibilité

Notre application se lance sur le localhost mais il est possible de configurer le port et l'adresse d'utilisation dans le document « application.properties » de notre application. Il se trouve dans « /src/main/java/resources/application.properties ».

```
springfox.documentation.swagger.v2.path=/api-docs
server.contextPath=/Alex57x/Projet/1.0.0
server.port=8082
spring.jackson.date-format=io.swagger.RFC3339DateFormat
spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS=false

spring.datasource.url=jdbc:h2:file:./devdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console
spring.h2.console.settings.trace=false
spring.h2.console.settings.web-allow-others=false

logging.level.org.hibernate.type=trace
logging.level.org.hibernate.sql=debug
```

Afin de changer le port, il suffit de changer la ligne « service.port » et pour changer l'adresse URL il suffit de changer la ligne « server.contextPath ». Par défaut, le projet est lancé sur l'adresse « http://localhost:8082/Alex57x/Projet/1.0.0 ».

## Documentations

Nous avons réalisé un Readme respectant certaines conventions qui permettent de mettre à disposition un grand nombre d'information. En plus, ces informations précédentes nous avons énormément de documentation sur le projet comme :

- Schéma de la base de données
- Verbose
- Carnet de bord
- Postman
- SwaggerEditor

Et la liste de toutes les technologies utilisées et de leur documentation pour le projet.

### 🔗 Démarrage

---

Afin de lancer le projet

- [Documentation](#) - Documentation
- [H2](#) - Base de données

### Fabriqué avec

---

Liste les programmes/logiciels/ressources que nous avons utilisé pour développer notre projet :

- [Swagger Editor](#) - Création de l'API
- [Spring Boot](#) - Framework
- [H2](#) - Base de données
- [Hibernate](#) - Framework pour le mapping
- [Lombok](#) - Générateur de code
- [Maven](#) - Pour les dépendances
- [Maven Docker](#) - Pour la gestion de maven dans le pipeline
- [Java 11](#) - Version du JAVA
- [Git](#) - Projet Git
- [Jacoco](#) - Coverage
- [Jacoco](#) - Maven jacoco

Pipeline d'intégration avec :

- [Gitea](#) - Build / Compile / Package
- [SonarQube](#) - Notation du code

### Documentation

---

- [Swagger Editor](#) - Document réalisé en Swagger
- [Carnet de bord](#) - Carnet de bord #A FAIRE#
- [Gantt](#) - Diagramme de Gantt #A FAIRE#
- [Postman](#) - Document Postman
- [BDD](#) - Schéma de la base de données
- [BDD\\_verbose](#) - Verbose de la base de données

### Versions

---

**Dernière version stable : 1.0.0**

Figure 1: La documentation du Readme

Cette capture d’écran permet de montrer l’exemple d’un schéma de BDD directement implanté dans notre documentation.

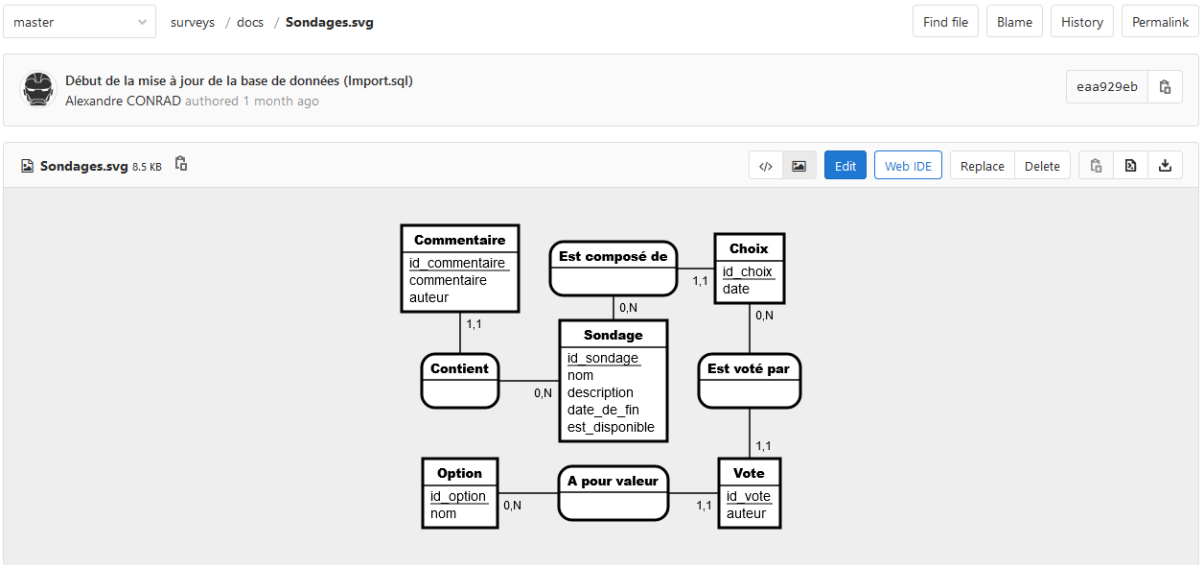


Figure 2: Capture d’écran de la base de données

### Comment utiliser notre projet

Pour utiliser notre application, deux possibilités s’offrent à vous.

La première consiste à utiliser la documentation offerte par Swagger lors du lancement du projet. Elle permet une prise en main facile et rapide de l’utilisation de l’application.

La deuxième consiste à l’utilisation de Postman, à l’aide du document à importer qui est disponible sur Git.

swagger

default (/api-docs)

Explore

### Projet M2 GI

Projet de production logiciel, réalisé par - BIR Vincent - CHEVALLET Clément - CONRAD ALEXandre - DAVIES Liam  
Projet à réaliser pour le xx / xx / 2020. Spécifications : - Java 11 - Rest API ( non obligatoirement RESTful) - Spring Boot - H2 pour la base de données - Mapping - Pipeline d'intégration - Documentations - Carnet de bord Notions du cours utilisé : - Swagger Editor - H2 - Hibernate - Lombok - Git

[Contact the developer](#)  
[Apache 2.0](#)

analytics	Show/Hide	List Operations	Expand Operations
choices	Show/Hide	List Operations	Expand Operations
comments	Show/Hide	List Operations	Expand Operations
surveys	Show/Hide	List Operations	Expand Operations
votes	Show/Hide	List Operations	Expand Operations

[ BASE URL: /Alex57x/Projet/1.0.0 , API VERSION: 1.0.0 ]

## Partie GET :

Pour utiliser notre application, il vous suffit d'utiliser le bouton « Try it out ! » de la documentation afin d'utiliser les différentes requêtes.

**surveys** [Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET /Alex57x/Projet/1.0.0/surveys [Retourne la liste de tous les sondages.](#)

**Implementation Notes**  
Retourne la liste de tous les sondages en cours ou qui ont eu lieu depuis la base de données.

**Response Class (Status 200)**  
Ressource trouvé

Model | **Example Value**

```
[
  {
    "description": "string",
    "endDate": {
      "date": 0,
      "day": 0,
      "hours": 0,
      "minutes": 0,
      "month": 0,
      "nanos": 0,
    }
  }
]
```

Response Content Type

**Response Messages**

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Ressource introuvable.		
500	Echec de connexion à la base de données.		

[Try it out!](#) [Hide Response](#)

Figure 3: "Try it out!" sur GET

L'utilisateur a le choix d'utiliser le CURL suggéré par la partie « Try it out ! », ou encore d'utiliser un client muni de la méthode « GET » pour appeler l'URL de la partie « Request URL ».

**Curl**

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys'
```

**Request URL**

```
http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys
```

**Request Headers**

```
{
  "Accept": "application/json"
}
```

Figure 4: Curl, l'url de requête et les headers de la requête



## Partie POST :

Les requêtes POST s'exécutent différemment des méthodes GET. De manière générale, nos méthodes POST servent à ajouter un jeu de données. Il faut donc se référer à la partie « Example Value » pour obtenir la liste des paramètres à envoyer.

Cependant, Swagger utilise un mauvais format pour le timestamp. Un utilisateur ne devra que se soucier d'envoyer un timestamp. Voici comment créer un survey avec la méthode POST de survey :

```
{
  "name": "Test survey",
  "description": "Création du survey !",
  "endDate": 1606669212000
}
```

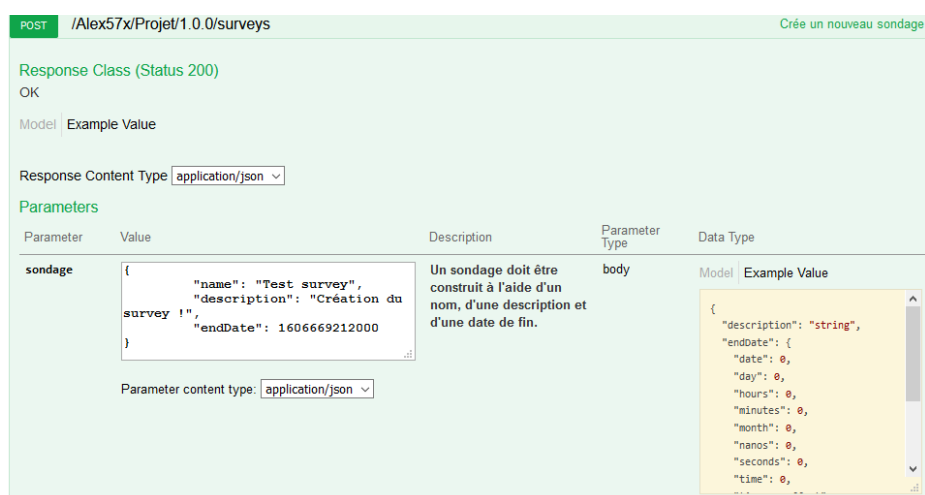


Figure 5: Création d'un sondage avec les valeurs données en exemple ci-dessus

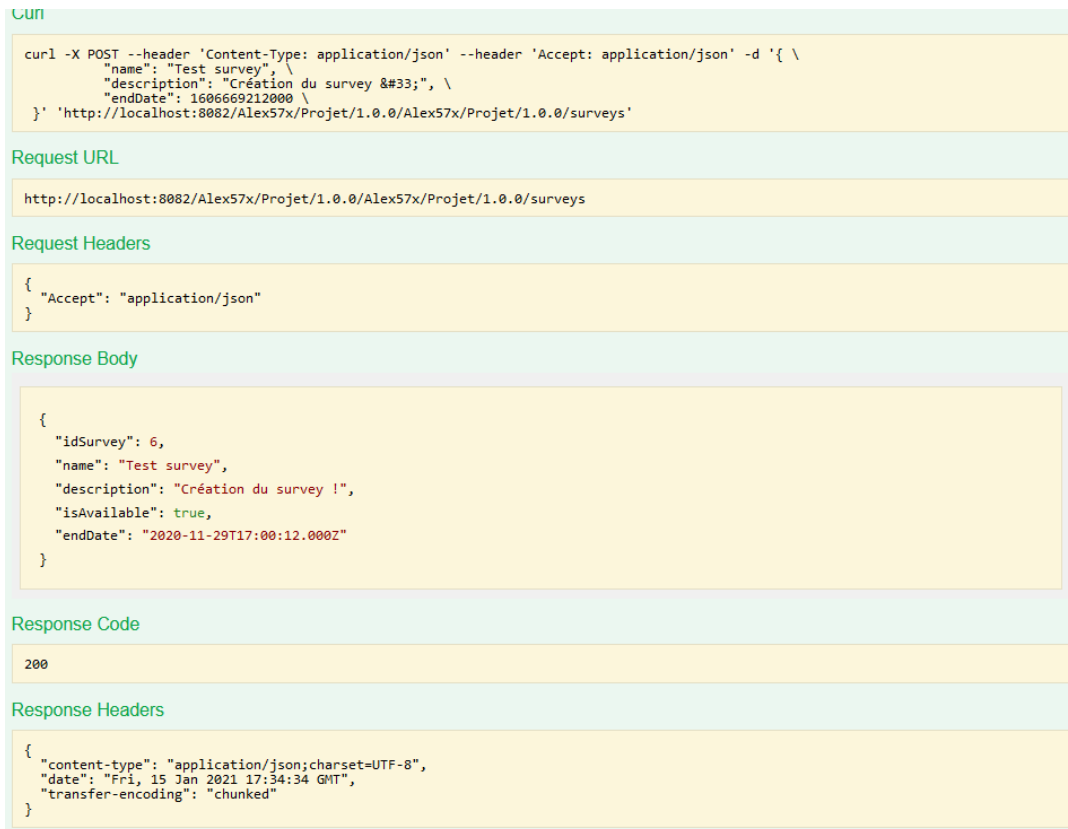


Figure 6: Exemple type d'un GET

## Partie PUT :

Les méthodes PUT sont très semblables aux méthodes POST, mais nécessitent généralement une information existante. Par exemple, modifier un sondage nécessite un id de sondage existant.

Pour modifier le survey ayant comme ID le numéro « 1 », il faudra donc appeler le lien suivant :

`http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/1`

Et le corps suivant :

```
{
  "name": "Nouveau nom de sondage",
  "description": "Nouvelle description",
  "isAvailable": true,
  "endDate": 1606669212000
}
```

#### Curl

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' -d ' \
{ \
  "name": "Nouveau nom de sondage", \
  "description": "Nouvelle description", \
  "isAvailable": true, \
  "endDate": 1606669212000 \
} \
' http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/1'
```

#### Request URL

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/1

#### Request Headers

```
{
  "Accept": "application/json"
}
```

#### Response Body

```
{
  "idSurvey": 1,
  "name": "Nouveau nom de sondage",
  "description": "Nouvelle description",
  "isAvailable": true,
  "endDate": "2020-11-29T17:00:12.000Z"
}
```

Figure 7: Exemple spécifique d'un POST

## Partie PATCH :

Notre seul appel API utilisant PATCH permet de clôturer un survey via son ID.

Le fonctionnement sera expliqué dans de plus amples détails dans la partie « [Faire des requêtes à notre API](#) ».



Figure 8: Exemple type d'un PATCH

## Partie DELETE :

Les appels utilisant la méthode DELETE permettent de supprimer un objet, ce qui est différent de clôturer un survey par exemple.

Par rapport au PATCH, seule la méthode d'appel change, autrement le fonctionnement est très similaire et seul l'ID d'un objet est demandé.

Curl
<pre>curl -X DELETE --header 'Accept: application/json' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/2'</pre>
Request URL
<pre>http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/2</pre>
Request Headers
<pre>{  "Accept": "application/json"}</pre>
Response Body
<pre>no content</pre>
Response Code
<pre>200</pre>
Response Headers
<pre>{  "content-length": "0",  "date": "Fri, 15 Jan 2021 17:50:20 GMT",  "content-type": null}</pre>

Figure 9: Exemple type d'un DELETE

## Postman

Nous avons mis à disposition un document pour pouvoir accéder à la documentation via Postman.

### Prérequis Postman

- Un compte Postman
- Le fichier contenant les infos à importer dans Postman (qui se situe dans « ./docs/Projet M2 Gl.postman\_collection.json »)

### Importation du fichier Postman

Une fois connecté, il faut accéder à l'onglet « home » puis cliquer sur « Import file » de la deuxième case nommée « Import an existing file ».

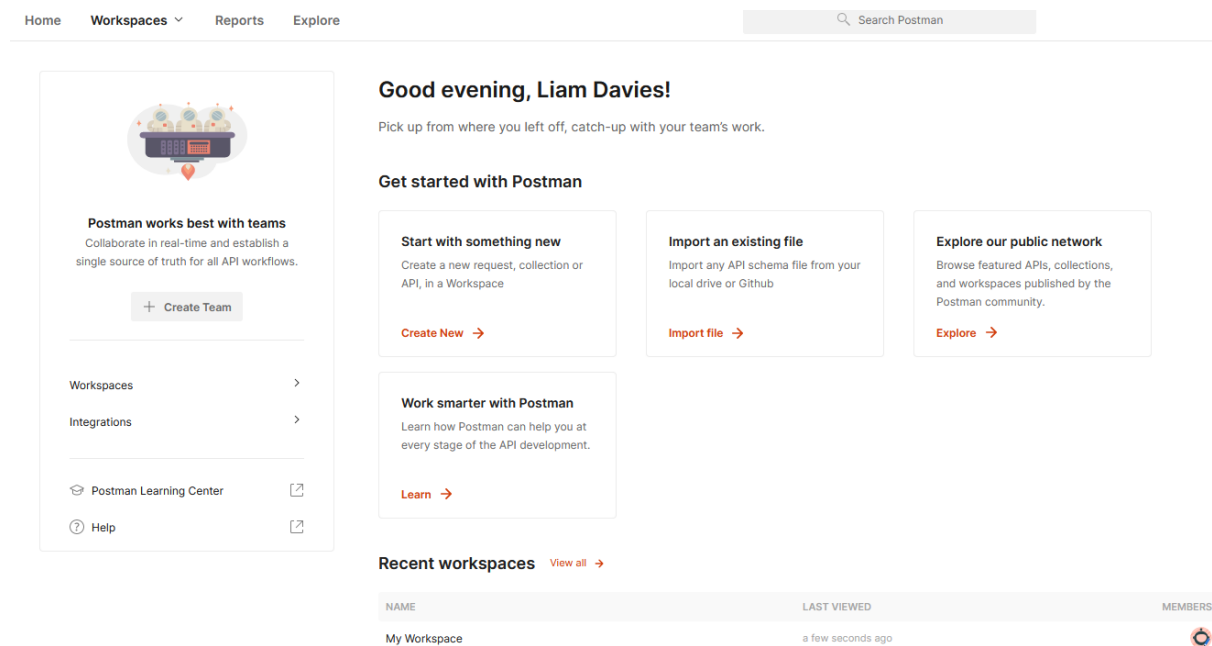


Figure 10: Interface Postman

Un nouvel onglet s'ouvre, il suffit de glisser déposer le fichier « Projet M2 Gl.postman\_collection.json » dedans et de cliquer sur « Import »

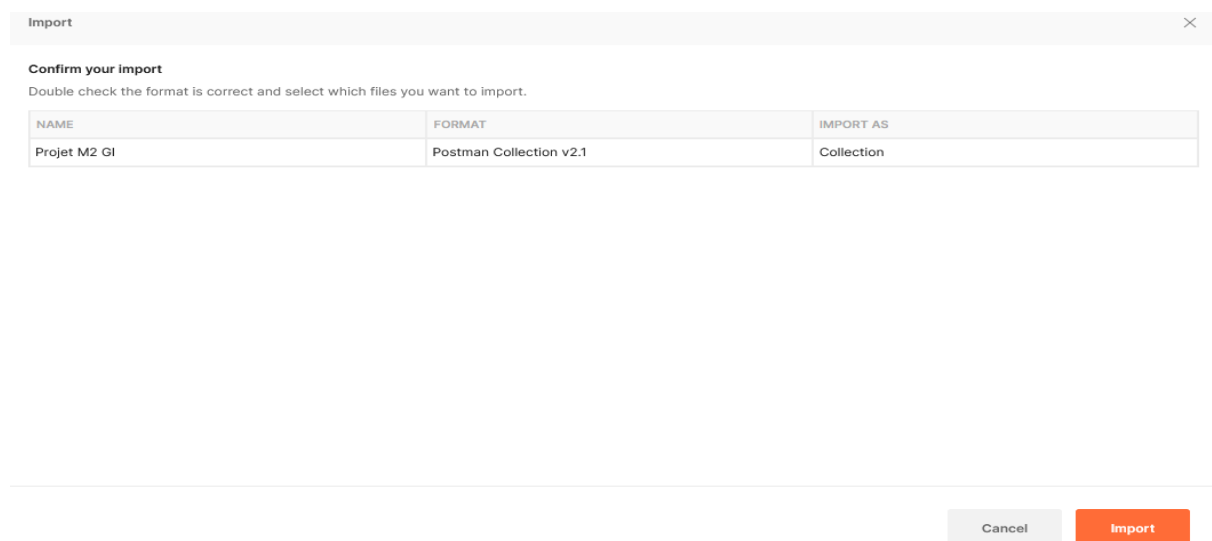
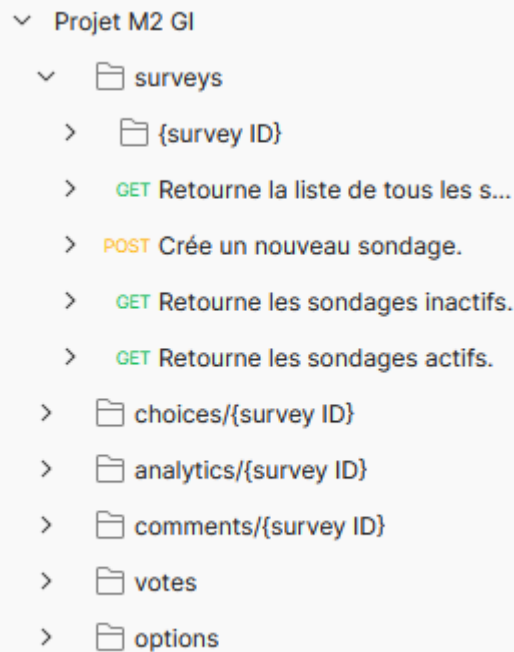


Figure 11: Onglet d'importation du fichier Postman.

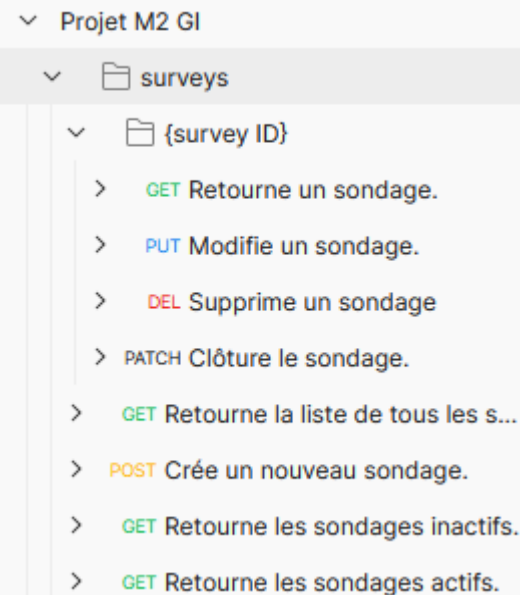
## Interface de Postman

Nous avons maintenant accès à la documentation de Postman, et une arborescence s'affiche en haut à gauche.

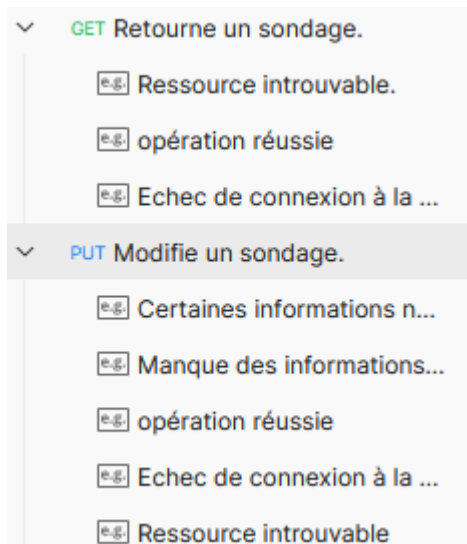
- 
- ▼ **Projet M2 GI**
    - ▼ **surveys**
      - > **{survey ID}**
      - > **GET** Retourne la liste de tous les s...
      - > **POST** Crée un nouveau sondage.
      - > **GET** Retourne les sondages inactifs.
      - > **GET** Retourne les sondages actifs.
    - > **choices/{survey ID}**
    - > **analytics/{survey ID}**
    - > **comments/{survey ID}**
    - > **votes**
    - > **options**

Nous retrouvons ici tous nos appels API. On remarque que « surveys » possède une sous-catégorie avec {survey ID}, cela signifie que tous les appels à « surveys » avec un paramètre nommé « survey ID » y sont rangés.

On retrouve donc dans ce premier dossier les méthodes GET et POST.

- 
- ▼ **Projet M2 GI**
    - ▼ **surveys**
      - ▼ **{survey ID}**
        - > **GET** Retourne un sondage.
        - > **PUT** Modifie un sondage.
        - > **DEL** Supprime un sondage
        - > **PATCH** Clôture le sondage.
      - > **GET** Retourne la liste de tous les s...
      - > **POST** Crée un nouveau sondage.
      - > **GET** Retourne les sondages inactifs.
      - > **GET** Retourne les sondages actifs.

Nous sommes descendus dans l'onglet {survey ID}, et nous retrouvons ici les méthodes demandant un ID de survey comme paramètre.



En descendant encore plus bas sur les 2 premières méthodes demandant un survey ID, nous avons un aperçu des status de retour.

En cliquant sur « **PUT** Modifie un sondage. », nous avons accès aux informations comme les paramètres ou le Body du jeu de données à envoyer.

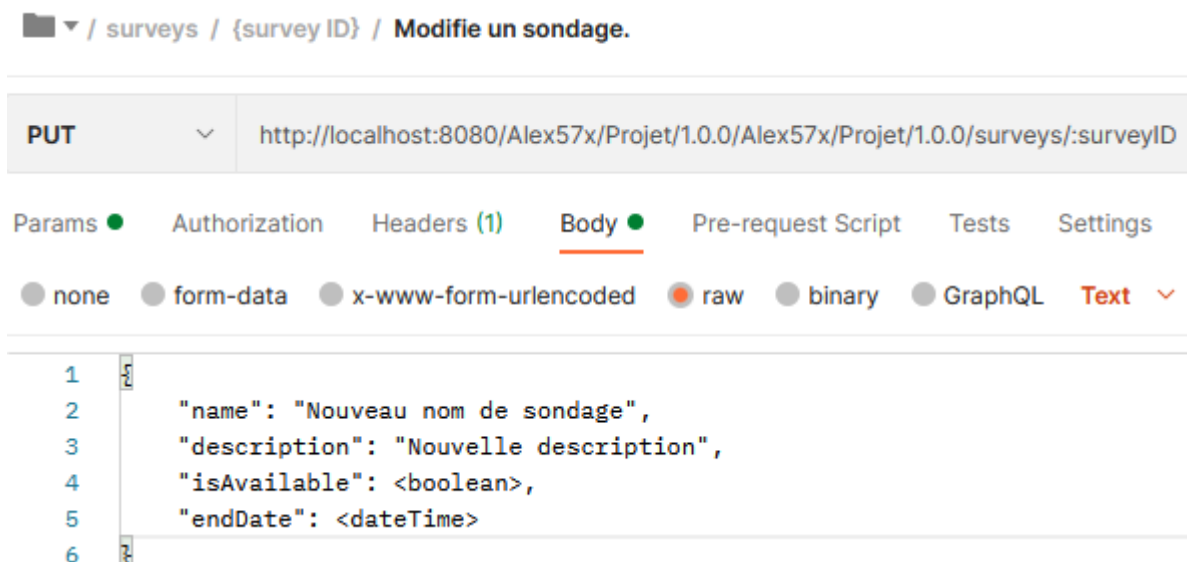


Figure 12: Un exemple de value à envoyer. Ici, le timestamp est affiché correctement.



## Faire des requêtes à notre API

Cette dernière partie va détailler de manière succincte et efficace les différents appels à notre API, en donnant un petit exemple si nécessaire.

### Date avec le plus de personnes présentes

GET /Alex57x/Projet/1.0.0/analytics/{surveyID}/available

Date avec le plus de personnes présentes

#### Paramètres :

- surveyID

#### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/analytics/surveyID/available'
```

#### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/analytics/surveyID/available

#### Réponse :

```
{
  "idChoice": 1,
  "date": "2021-01-31T11:00:00.000Z",
  "idSurvey": surveyID
}
```

---

### Date avec le plus de personnes potentiellement présentes

GET /Alex57x/Projet/1.0.0/analytics/{surveyID}/maybe

Date avec le plus de personnes potentiellement présentes

#### Paramètres :

- surveyID

#### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/analytics/surveyID/maybe'
```

#### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/analytics/surveyID/maybe

#### Réponse :

```
{
  "idChoice": 1,
  "date": "2021-01-31T11:00:00.000Z",
  "idSurvey": surveyID
}
```

---

## Supprimer un choix

DELETE

/Alex57x/Projet/1.0.0/choices/{choiceID}

Supprime un choix

### Paramètres :

- choiceID

### Curl :

```
curl -X GET --header 'Accept: application/json' curl -X DELETE --header 'Accept: application/json' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID

### Réponse :

no content

---

## Afficher la liste des choix possibles

GET

/Alex57x/Projet/1.0.0/choices/{surveyID}

Liste des choix possibles

### Paramètres :

- choiceID

### Curl :

```
curl -X GET --header 'Accept: application/json' curl -X DELETE --header 'Accept: application/json' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID

### Réponse :

```
[
  {
    "idChoice": 1,
    "date": "2021-01-31T11:00:00.000Z",
    "idSurvey": 1
  },
  {
    "idChoice": 2,
    "date": "2020-12-25T11:00:00.000Z",
    "idSurvey": 1
  }
] (Un tableau contenant les choix)
```

---

## Ajouter un choix pour un sondage

POST

/Alex57x/Projet/1.0.0/choices/{surveyID}

Ajouter un choix pour un sondage

### Paramètres :

- choice (timestamp)
- surveyID

### Curl :

```
curl -X GET --header 'Accept: application/json' curl -X DELETE --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/choices/choiceID

### Réponse :

```
{
  "idChoice": 9,
  "date": "1970-01-19T15:25:41.864Z",
  "idSurvey": 3
}
```

---

## Récupérer tous les commentaires d'un sondage

GET

/Alex57x/Projet/1.0.0/comments/{surveyID}

Récupère tous les commentaires

### Paramètres :

- surveyID

### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/comments/surveyID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/comments/surveyID

### Réponse :

```
[
  {
    "idComment": 1,
    "comment": "Je serai là !",
    "author": "Liam",
    "idSurvey": 1
  }
] (un array de commentaires)
```

---

## Ajouter un commentaire sur un sondage

POST

/Alex57x/Projet/1.0.0/comments/{surveyID}/{auteur}

Ajoute un commentaire

### Paramètres :

- surveyID
- auteur (string)
- message (string)

### Curl :

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d 'J%27y  
serai' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/comments/surveyID/Liam'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/comments/surveyID/Liam

### Réponse :

```
{  
  "idComment": 5,  
  "comment": "J'y serai",  
  "author": "Liam",  
  "idSurvey": 1  
}
```

---

## Récupérer la liste de tous les sondages

GET

/Alex57x/Projet/1.0.0/surveys

Retourne la liste de tous les sondages.

### Paramètres : Aucun

### Curl :

```
curl -X GET --header 'Accept: application/json'  
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys

### Réponse :

```
[  
  {  
    "idSurvey": 1,  
    "name": "Anniversaire surprise pour Alexandre ?",  
    "description": "On fait une surprise, ne lui dites pas !!",  
    "isAvailable": true,  
    "endDate": "2020-12-31T11:00:00.000Z"  
  },  
  {  
    "idSurvey": 2,  
    "name": "Projet X après le déconfinement ?",  
    "description": "Ça va être mortel !!",  
    "isAvailable": false,  
  }  
]
```

```

    "endDate": "2020-12-31T11:00:00.000Z"
  },
  {
    "idSurvey": 3,
    "name": "Soirée Netflix & Chill ?",
    "description": "On regardera Star Wars :)",
    "isAvailable": true,
    "endDate": "2021-01-31T11:00:00.000Z"
  },
  {
    "idSurvey": 4,
    "name": "Tour des garages pour trouver ma nouvelle voiture ?",
    "description": "Je veux une lambo, minimum",
    "isAvailable": true,
    "endDate": "2021-01-31T11:00:00.000Z"
  }
]

```

---

### Créer un nouveau sondage

POST /Alex57x/Projet/1.0.0/surveys

Crée un nouveau sondage.

### Paramètres :

- Sondage (construit à partir d'un nom, d'une description et d'un timestamp)

### Par exemple :

```

{
  "name": "Test update",
  "description": "update sondage !",
  "endDate": 1606669212000
}

```

### Curl :

```

curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
    "name": "Sondage présentation", \
    "description": "Qui sera là pour la présentation du projet Spring \
?", \
    "endDate": 1606669212000 \
}'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys'

```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys

### Réponse :

```

{
  "idSurvey": 5,
  "name": "Sondage présentation",
  "description": "Qui sera là pour la présentation du projet Spring ?",
  "isAvailable": true,
  "endDate": "2020-11-29T17:00:12.000Z"
}

```

---

## Récupérer la liste des sondages actifs

GET /Alex57x/Projet/1.0.0/surveys/actives

[Retourne les sondages actifs.](#)

**Paramètres :** Aucun

### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/actives'
```

### Requête HTTP :

`http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/actives`

### Réponse :

```
[
  {
    "idSurvey": 1,
    "name": "Anniversaire surprise pour Alexandre ?",
    "description": "On fait une surprise, ne lui dites pas !!",
    "isAvailable": true,
    "endDate": "2020-12-31T11:00:00.000Z"
  },
  {
    "idSurvey": 3,
    "name": "Soirée Netflix & Chill ?",
    "description": "On regardera Star Wars :)",
    "isAvailable": true,
    "endDate": "2021-01-31T11:00:00.000Z"
  },
  {
    "idSurvey": 4,
    "name": "Tour des garages pour trouver ma nouvelle voiture ?",
    "description": "Je veux une lambo, minimum",
    "isAvailable": true,
    "endDate": "2021-01-31T11:00:00.000Z"
  },
  {
    "idSurvey": 5,
    "name": "Sondage présentation",
    "description": "Qui sera là pour la présentation du projet Spring ?",
    "isAvailable": true,
    "endDate": "2020-11-29T17:00:12.000Z"
  }
] (Un tableau de réponses)
```

---

## Récupérer la liste des sondages inactifs

GET /Alex57x/Projet/1.0.0/surveys/expireds

[Retourne les sondages inactifs.](#)

**Paramètres :** Aucun

### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/expireds'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/expired

### Réponse :

```
[
  {
    "idSurvey": 2,
    "name": "Projet X après le déconfinement ?",
    "description": "Ça va être mortel !!",
    "isAvailable": false,
    "endDate": "2020-12-31T11:00:00.000Z"
  }
] (Un tableau de réponses)
```

---

### Supprimer un sondage

DELETE	/Alex57x/Projet/1.0.0/surveys/{surveyID}	Supprime un sondage
--------	--	---------------------

### Paramètres :

- surveyID

### Curl :

```
curl -X DELETE --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID

### Réponse :

Aucune réponse.

---

### Récupérer un sondage spécifique

GET	/Alex57x/Projet/1.0.0/surveys/{surveyID}	Retourne un sondage.
-----	--	----------------------

### Paramètres :

- surveyID

### Curl :

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID

### Réponse :

```
{
  "idSurvey": 1,
  "name": "Anniversaire surprise pour Alexandre ?",
  "description": "On fait une surprise, ne lui dites pas !!",
  "isAvailable": true,
  "endDate": "2020-12-31T11:00:00.000Z"
}
```

### Clôturer un sondage avant la date de fin

PATCH

/Alex57x/Projet/1.0.0/surveys/{surveyID}

Clôture le sondage.

### Paramètres :

- surveyID

### Curl :

```
curl -X PATCH --header 'Content-Type: application/json' --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/1'
```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/1

### Réponse :

Aucune réponse

---

### Modifier les informations d'un sondage

PUT

/Alex57x/Projet/1.0.0/surveys/{surveyID}

Modifie un sondage.

### Paramètres :

- surveyID
- body (demande une description (string), une endDate (timestamp), un isAvailable (boolean) et un name (string))

Exemple :

```
{
  "name": "string",
  "description": "string",
  "isAvailable": true,
  "endDate": 1606669212000
}
```

### Curl :

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "description": "string", \
  "endDate": 1606669212000, \
  "isAvailable": true, \
  "name": "string" \
}
```



```
}'  
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID'
```

### **Requête HTTP :**

<http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/surveys/surveyID>

### **Réponse:**

```
{  
  "idSurvey": 3,  
  "name": "string",  
  "description": "string",  
  "isAvailable": true,  
  "endDate": "2020-11-29T17:00:12.000Z"  
}
```

---

### Récupérer toutes les options

GET

/Alex57x/Projet/1.0.0/option/

Retourne toutes les options

**Paramètres :** Aucun

### **Curl :**

```
curl -X GET --header 'Accept: application/json'  
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/option/'
```

### **Requête HTTP :**

<http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/option/>

### **Réponse :**

```
[  
  {  
    "idOption": 1,  
    "name": "Disponible"  
  },  
  {  
    "idOption": 2,  
    "name": "Indisponible"  
  },  
  {  
    "idOption": 3,  
    "name": "Peut-être"  
  }  
] (Un tableau d'options)
```

---

## Ajout d'une option

POST

/Alex57x/Projet/1.0.0/option/

Ajout d'une option

### **Paramètres :**

- optionName (string)

### **Exemple :**

Nouvelle option

### **Curl :**

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d 'Test' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/option/'
```

### **Requête HTTP :**

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/option/

### **Réponse :**

```
{
  "idOption": 4,
  "name": "Test"
}
```

---

## Récupérer tous les votes d'un choix

GET

/Alex57x/Projet/1.0.0/votes/{choiceID}/choice/

Retourne tous les votes d'un choix

### **Paramètres :**

- choiceID

### **Curl :**

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/choiceID/choice/'
```

### **Requête HTTP :**

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/choiceID/choice/

### **Réponse :**

```
[
  {
    "idVote": 10,
    "author": "Liam",
    "idChoice": 3,
    "idOption": 1
  },
  {
    "idVote": 12,
    "author": "Alex",
    "idChoice": 3,
    "idOption": 3
  },
  {
    "idVote": 14,
    "author": "Vincent",
    "idChoice": 3,
    "idOption": 1
  },
  {
    "idVote": 16,
    "author": "Clément",
    "idChoice": 3,
    "idOption": 2
  }
]
```

---

### Récupérer tous les votes d'une option

GET

/Alex57x/Projet/1.0.0/votes/{optionID}/option/

[Retourne tous les votes d'une option](#)

### **Paramètres :**

- optionID

### **Curl :**

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/opti
onID/option/'
```

### **Requête HTTP :**

```
http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/optio
nID/option/
```

### **Réponse :**

```
[
  {
    "idVote": 10,
    "author": "Liam",
    "idChoice": 3,
    "idOption": 1
  },
  {
    "idVote": 11,
    "author": "Alexandre",
    "idChoice": 4,
```

```

        "idOption": 1
    },
    {
        "idVote": 14,
        "author": "Vincent",
        "idChoice": 3,
        "idOption": 1
    },
    {
        "idVote": 15,
        "author": "Vincent",
        "idChoice": 4,
        "idOption": 1
    },
    {
        "idVote": 18,
        "author": "Clément",
        "idChoice": 5,
        "idOption": 1
    },
    {
        "idVote": 19,
        "author": "Clément",
        "idChoice": 6,
        "idOption": 1
    }
]

```

---

## Participation d'un utilisateur

POST

/Alex57x/Projet/1.0.0/votes/{optionID}/{choiceID}

Participer à un sondage

### Paramètres :

- optionID
- choiceID
- auteur (string)

### Curl :

```

curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d 'Liam' 'http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/optionID/choiceID'

```

### Requête HTTP :

http://localhost:8082/Alex57x/Projet/1.0.0/Alex57x/Projet/1.0.0/votes/optionID/choiceID

### Réponse :

```

{
    "idVote": 28,
    "author": "Liam",
    "idChoice": 1,
    "idOption": 1
}

```