

# DunPepene's documentation

## by IOPETI Hugo & YVOZ Ludovic

Nous avons visualisé ce projet comme étant une application web permettant de gérer des agendas de la pâtisserie "DunPepene".

Selon notre implémentation, nous pouvons faire évoluer cette application pour répondre aux différents besoins de ce commerce.

<b>Mise en place du projet</b>	<b>2</b>
Lancement du serveur local	2
Obtention du client Web	2
<b>Documentation de l'API</b>	<b>2</b>
Présentation	2
Opérations	4
Auth	4
SignIn	5
SignUp	7
GetUsername	8
Calendar	9
Month	9
Week	10
Day	11
Events	12
All	12
GetEvent	13
Edit	14
Add	15
Delete	16
Files	17
Index	17
Calendar	18
CalendarShowMonth	19
CalendarShowWeek	20
CalendarShowDay	21
Login	22
Register	23
Specials	24
Posts	24
401	25
500	26
Teapot	27
404	28

# Mise en place du projet

## Lancement du serveur local

Pour lancer le serveur, il faut exécuter les commandes suivantes :

```
> npm install  
> npm run-script run
```

## Obtention du client Web

Pour accéder aux différentes pages de notre site, il suffit d'écrire "https://localhost:3030" suivi des différentes routes.

# Documentation de l'API

## Présentation

Notre API est utilisable extérieurement via curl. Lors de notre projet, nous avons utilisé l'application Postman qui est basée sur curl pour tester nos opérations.

Nos opérations sont regroupées sous 5 catégories :

- Auth
- Calendar
- Event
- Files
- Specials

## Collections



APIs



Environments



Mock Servers



Monitors



Flows



History



...

## Auth

POST SignIn

POST SignUp

GET GetUsername

## Calendar

GET Month

GET Week

GET Day

## Events

GET GetAllEvents

GET GetEvent

POST Edit

POST Add

DEL Delete

## Files

GET Register

GET Login

GET Index

GET Calendar

GET CalendarShowMonth

GET CalendarShowWeek

GET CalendarShowDay

## Specials

GET 401

GET 500

GET Teapot

GET 404

## Opérations

Les captures d'écrans qui accompagnent les descriptions sont des captures d'écran du logiciel postman (les {{LH}} sont des variables représentant "localhost:3030/").

### **Auth**

La catégorie Auth regroupe toutes les opérations permettant d'interagir avec le système de comptes de l'application, ces opérations sont décrites ci-dessous.

## SignIn

SignIn est une requête POST qui permet de demander à l'API de se connecter à l'application. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/signin.
- Le body de l'opération (encodé en x-www-form-urlencoded) est le suivant :
  - email (représentant une adresse mail)
  - password (représentant un mot de passe en clair)
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Un token jwt permettant l'authentification de l'utilisateur dans l'application si la requête à abouti sans erreur (status : 200).

Sur le site <https://jwt.io/>, nous pouvons voir comment est formé notre token, dans la partie payload, soit le contenu du token, on peut voir que l'on a décidé de transmettre l'email d'un utilisateur (clé/identifiant unique est propre à l'utilisateur étant donné qu'il ne peut y avoir qu'un seul compte par mail). IAT correspond à la date de création du token, soit à la date à laquelle un utilisateur s'est inscrit/connecté. EXP correspond à la date d'expiration du token, fixée à 1h.

### Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imh1Z29AZGluZG9uLmZyIiwiaWF0IjoxNjY3NDk1MTI3LCJleHAiOjE2Njc0OTg3Mjd9.t8Qu4g1H6VABNP0fAUw6o_cT7xAfiXL401cer80Ai-E
```

### Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "email": "hugo@dindon.fr",
  "iat": 1667495127,
  "exp": 1667498727
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

- La chaîne de caractère "failure" si l'authentification est impossible car les données passées dans le body ne correspondent pas aux informations stockées dans notre fichier .json (status : 400).
  - La chaîne de caractère "error" si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - email = hugo.dindon.fr
  - password = Dindon54

POST

{{LH}}signin

Send

Params

Auth

Headers (8)

Body

Pre-req.

Tests

Settings

Cookies

x-www-form-urlencoded

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	hugo@dindon.fr			
<input checked="" type="checkbox"/>	password	Dindon54			
	Key	Value	Description		

Body

200 OK 14 ms 415 B Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFnZW50IjZ29AZGluZG9uLmZyIiwiaWF0IjoxNjY3NDE2MjA3LCJleHAiOjE2Njc0MTk4MDd9.EEs8vhU7ZZHDbFhRiCeo0iTRZFie4PEVQtigZs4r7Cg"



## GetUsername

GetUsername est une requête GET qui permet de demander à l'API de nous renvoyer le nom d'utilisateur correspondant à une adresse mail. Le format de la requête est le suivant :

- La requête se formule comme cela :  
`localhost:3030/account/getUsername?email="XXX@YY.Z"`.
  - Le seul paramètre est l'email.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le nom d'utilisateur au format texte si la requête à abouti sans problème (status : 200).
  - La chaîne de caractère "failure" si l'email envoyé en paramètre ne correspond pas à un compte dans le fichier .json (status : 400).
  - La chaîne de caractère "error" si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - email = hugo@dindon.fr

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

- Method:** GET
- URL:** `{{LH}}account/getUsername?email=hugo@dindon.fr`
- Buttons:** Send, Cookies
- Params:** Auth, Headers (6), Body, Pre-req., Tests, Settings
- Query Params:**

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	hugo@dindon.fr			
	Key	Value	Description		
- Body:** 200 OK, 4 ms, 230 B, Save Response
- Response Format:** Pretty, Raw, Preview, Visualize, HTML
- Response Content:** 1 Hugo



## Calendar

La catégorie Calendar regroupe toutes les opérations permettant de récupérer les événements à afficher sur le calendrier de l'utilisateur à un moment donné. Ces opérations sont décrites ci-dessous.

### Month

Month est une requête GET qui permet de demander à l'API de nous renvoyer tous les événements d'un utilisateur sur une durée d'un mois. Le format de la requête est le suivant :

- La requête se formule comme cela :  
`localhost:3030/calendar/month?date="XXXX-YY-ZZ&email=AAA@BB.C"`.
  - Le premier paramètre est la date que l'utilisateur est en train de consulter, XXXX correspond à l'année, YY correspond au mois et ZZ correspond au jour.
  - Le second paramètre est l'email de l'utilisateur dont on veut les événements.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Les événements en texte mais formatés comme un fichier json (status : 200).
  - Les caractères "[]" si aucun événement n'est trouvé en cas de paramètres incorrects ou tout simplement dû au fait qu'il n'y ait aucun événement à renvoyer(status : 200).
  - Une erreur d'impossibilité de GET si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - date = 2022-10-29
  - email = hugo@dindon.fr

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

- Method:** GET
- URL:** `{{LH}}calendar/month?date=2022-10-22&email=hugo@dindon.fr`
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, Cookies
- Query Params:**

KEY	VALUE	DESCRIPTION
date	2022-10-22	
email	hugo@dindon.fr	
- Body:** Cookies, Headers (7), Test Results
- Status:** 200 OK, 13 ms, 2 KB
- Response Format:** Pretty, Raw, Preview, Visualize, JSON
- Response Body (JSON):**

```
[
  {
    "id": "0",
    "owner": "Hugo",
    "owner_email": "hugo@dindon.fr",
    "title": "Ingredients",
    "description": "Acheter des ingrédients frais au marché.",
    "date": "2022-10-29",
    "duration": "2",
    "start_time": "9",
    "color": "red"
  },
  {
    "id": "1",
```

## Week

Week est une requête GET qui permet de demander à l'API de nous renvoyer tous les événements d'un utilisateur sur une durée d'une semaine. Le format de la requête est le suivant :

- La requête se formule comme cela :  
`localhost:3030/calendar/week?date="XXXX-YY-ZZ&email=AAA@BB.C"`.
  - Le premier paramètre est la date que l'utilisateur est en train de consulter, XXXX correspond à l'année, YY correspond au mois et ZZ correspond au jour.
  - Le second paramètre est l'email de l'utilisateur dont on veut les événements.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Les événements en texte mais formatés comme un fichier json (status : 200).
  - Les caractères "[]" si aucun événement n'est trouvé en cas de paramètres incorrects ou tout simplement dû au fait qu'il n'y ait aucun événement à renvoyer(status : 200).
  - Une erreur d'impossibilité de GET si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - date = 2022-10-29
  - email = hugo@dindon.fr

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

- Method:** GET
- URL:** `{{LH}}calendar/week?date=2022-10-29&email=hugo@dindon.fr`
- Params:** date=2022-10-29, email=hugo@dindon.fr
- Status:** 200 OK, 23 ms, 1.4 KB
- Response Body (JSON):**

```
[
  {
    "id": "0",
    "owner": "Hugo",
    "owner_email": "hugo@dindon.fr",
    "title": "Ingredients",
    "description": "Acheter des ingrédients frais au marché.",
    "date": "2022-10-29",
    "duration": "2",
    "start_time": "9",
    "color": "red"
  },
  {
    "id": "3",
```

## Day

Day est une requête GET qui permet de demander à l'API de nous renvoyer tous les événements d'un utilisateur sur une durée d'un jour. Le format de la requête est le suivant :

- La requête se formule comme cela :  
`localhost:3030/calendar/day?date="XXXX-YY-ZZ&email=AAA@BB.C"`.
  - Le premier paramètre est la date que l'utilisateur est en train de consulter, XXXX correspond à l'année, YY correspond au mois et ZZ correspond au jour.
  - Le second paramètre est l'email de l'utilisateur dont on veut les événements.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Les événements en texte mais formatés comme un fichier json (status : 200).
  - Les caractères "[]" si aucun événement n'est trouvé en cas de paramètres incorrects ou tout simplement dû au fait qu'il n'y ait aucun événement à renvoyer(status : 200).
  - Une erreur d'impossibilité de GET si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - `date = 2022-10-29`
  - `email = hugo@dindon.fr`

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

- Method:** GET
- URL:** `{{LH}}calendar/day?date=2022-10-29&email=hugo@dindon.fr`
- Status:** 200 OK, 6 ms, 1.03 KB
- Response Format:** JSON
- Response Body:**

```
1 []
2 {
3   "id": "0",
4   "owner": "Hugo",
5   "owner_email": "hugo@dindon.fr",
6   "title": "Ingredients",
7   "description": "Acheter des ingrédients frais au marché.",
8   "date": "2022-10-29",
9   "duration": "2",
10  "start_time": "9",
11  "color": "red"
12 },
13 {
14  "id": "3",
```

## Events

La catégorie Events regroupe toutes les opérations permettant d'interagir avec les événements. Ces opérations sont décrites ci-dessous.

### All

All est une requête GET qui permet de demander à l'API de nous renvoyer tous les événements en mémoire dans l'application. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/events.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Les événements en texte mais formatés comme un fichier json (status : 200).
  - Les caractères "[]" si aucun événement n'est trouvé en cas de paramètres incorrects ou tout simplement dû au fait qu'il n'y ait aucun événement à renvoyer(status : 200).
  - Une erreur d'impossibilité de GET si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

The screenshot shows a REST client interface with a GET request to `{{LH}}events`. The response is a JSON array of event objects. The status is 200 OK, with a response time of 16 ms and a size of 2 KB.

**Query Params**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

**Body** 200 OK 16 ms 2 KB Save Response

**Pretty** **Raw** **Preview** **Visualize** **JSON** 🔍

```
1 [
2   {
3     "id": "0",
4     "owner": "Hugo",
5     "owner_email": "hugo@dindon.fr",
6     "title": "Ingredients",
7     "description": "Acheter des ingrédients frais au marché.",
8     "date": "2022-10-29",
9     "duration": "2",
10    "start_time": "9",
11    "color": "red"
12  },
13  {
14    "id": "1",
```

## GetEvent

GetEvent est une requête GET qui permet de demander à l'API de nous renvoyer le nom d'utilisateur correspondant à une adresse mail. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/account/events/get?id="X".
  - Le seul paramètre est l'id de l'événement à récupérer.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - L'événement en texte mais formaté comme un fichier json (status : 200).
  - Aucune réponse si aucun événement n'a été trouvé (status : 204).
  - Une erreur d'impossibilité de GET si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - id=2

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

- Method:** GET
- URL:** {{LH}}events/get?id=2
- Send Button:** A blue button labeled "Send".
- Params Tab:** Active, showing "Query Params".
- Query Params Table:**

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	2			
	Key	Value	Description		
- Body Tab:** Active, showing the response body.
- Status Bar:** 200 OK, 6 ms, 428 B, with a "Save Response" button.
- Response Format:** Pretty (JSON).
- Response Body (JSON):**

```
1 {
2   "id": "2",
3   "owner": "Hugo",
4   "owner_email": "hugo@dindon.fr",
5   "title": "Fromage",
6   "description": "Chercher le fromage dans la cave.",
7   "date": "2022-10-22",
8   "duration": "4",
9   "start_time": "9",
10  "color": "green"
11 }
```

## Edit

Edit est une requête POST qui permet de demander à l'API de modifier un événement dans le fichier json. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/events/edit.
- Le body de l'opération (encodé en x-www-form-urlencoded) est le suivant :
  - id (représentant l'id de l'événement à modifier)
  - Les champs de l'événement après modification (ou non) :
    - owner, owner\_email, title, description, date, duration, start\_time, color
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - La chaîne de caractère "success" si la requête à abouti sans erreur (status : 201).
  - La chaîne de caractère "failure" si l'id ne correspond pas à un événement ou si le champ owner-email à changé (status : 400).
  - La chaîne de caractère "error" si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - id = 666, owner = Hugo, owner-email = [hugo@dindon.fr](mailto:hugo@dindon.fr), title = Cuisine, description = Travailler en cuisine, date = 2022-10-23, duration = 6, start\_time = 14, color = orange

The screenshot shows a REST client interface with a POST request to `{{LH}}events/edit`. The body is encoded as `x-www-form-urlencoded` and contains the following data:

Field	Value
id	666
owner	Hugo
owner_email	hugo@dindon.fr
title	Cuisine
description	Travailler en cuisine
date	2022-10-23
duration	6
start_time	14
color	orange

The response status is 201 Created, with a response time of 12 ms and a size of 238 B. The response body is `1 success`.

## Add

Add est une requête POST qui permet de demander à l'API d'ajouter un événement dans le fichier json. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/events/add.
- Le body de l'opération (encodé en x-www-form-urlencoded) est le suivant :
  - Les champs de l'événement :
    - id, owner, owner\_email, title, description, date, duration, start\_time, color
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - La chaîne de caractère "success" si la requête à abouti sans erreur (status : 201).
  - La chaîne de caractère "failure" si l'id est déjà utilisé par un événement existant (status : 400).
  - La chaîne de caractère "error" si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - id = 667, owner = Hugo, owner-email = [hugo@dindon.fr](mailto:hugo@dindon.fr), title = Vaisselle, description = Faire la vaisselle à nouveau, date = 2022-10-24, duration = 6, start\_time = 14, color = orange

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

- Method:** POST
- URL:** {{LH}}events/add
- Body Type:** x-www-form-urlencoded
- Body Data:**

Field	Value
id	667
owner	Hugo
owner_email	hugo@dindon.fr
title	Vaisselle
description	Faire la vaisselle à nouveau
date	2022-10-23
duration	4
start_time	14
color	orange
- Response:** 201 Created, 17 ms, 238 B
- Response Body:** 1 success

## Delete

Delete est une requête POST qui permet de demander à l'API de supprimer un événement dans le fichier json. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/events/delete.
- Le body de l'opération (encodé en x-www-form-urlencoded) est le suivant :
  - id (représentant l'id de l'événement à supprimer)
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Aucune réponse si la requête à abouti sans erreur (status : 204).
  - La chaîne de caractère "failure" si l'id ne correspond pas à un événement (status : 400).
  - La chaîne de caractère "error" si la requête à été interrompue par n'importe quelle autre raison (status 400).
- Un exemple fonctionnel est d'envoyer une requête ayant comme données :
  - id = 666

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

- Method:** DELETE
- URL:** {{LH}}events/delete
- Body Type:** x-www-form-urlencoded
- Body Content:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	666	
Key	Value	Description
- Response:** 204 No Content, 43 ms, 134 B
- Response Format:** Pretty



## Files

### Index

Index est une requête GET qui permet de demander à l'API de nous renvoyer le fichier index.html qui correspond à la fenêtre d'accueil. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/index.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

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

- Method:** GET
- URL:** {{LH}}index
- Status:** 200 OK
- Response Time:** 32 ms
- Response Size:** 1.1 KB
- Body:** HTML (Pretty view)

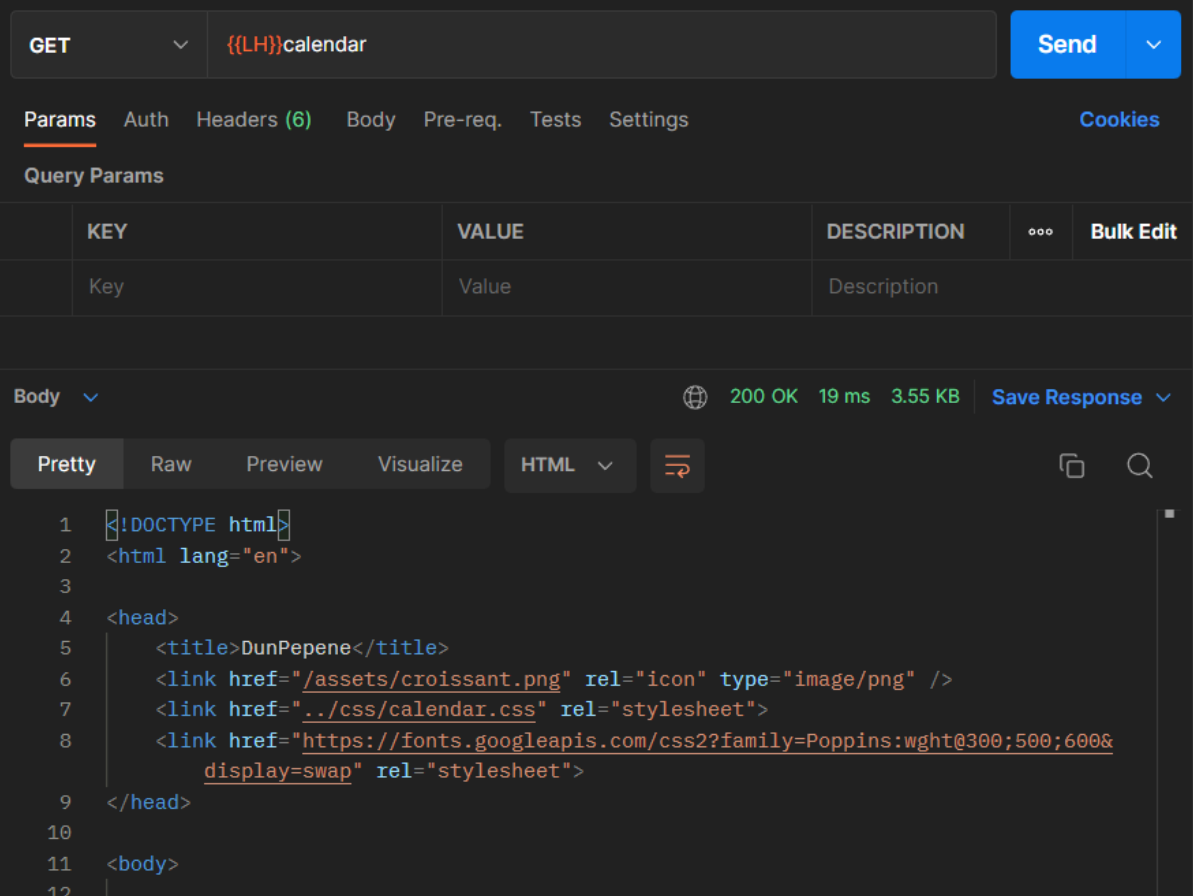
The response body contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <title>DunPepene's Homepage</title>
6   <link href="/assets/croissant.png" rel="icon" type="image/png" />
7   <link href="css/style.css" rel="stylesheet">
8   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
  display=swap" rel="stylesheet">
9 </head>
10
11 <body>
```

## Calendar

Calendar est une requête GET qui permet de demander à l'API de nous renvoyer le fichier `calendarView.html` qui correspond à la fenêtre affichant le calendrier. Le format de la requête est le suivant :

- La requête se formule comme cela : `localhost:3030/calendar`.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.



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

- Method:** GET
- URL:** `{{LH}}calendar`
- Status:** 200 OK
- Time:** 19 ms
- Size:** 3.55 KB
- Response Format:** HTML

The response body is an HTML document with the following structure:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <title>DunPepene</title>
6   <link href="/assets/croissant.png" rel="icon" type="image/png" />
7   <link href="../css/calendar.css" rel="stylesheet">
8   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
  display=swap" rel="stylesheet">
9 </head>
10
11 <body>
12
```

## CalendarShowMonth

CalendarShowMonth est une requête GET qui permet de demander à l'API de nous renvoyer le fichier calendarMonthView.html qui correspond à la fenêtre affichant le calendrier dans sa version "month". Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/calendar/show/month.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

GET `{{LH}}calendar/show/month` Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 21 ms 2.99 KB Save Response

Pretty Raw Preview Visualize HTML

```
1 <div class="frame">
2   <table align="center" cellpadding="21" cellspacing="21" class="calendarMonth">
3     <caption align="top">
4     </caption>
5     <thead>
6     <tr>
7       <th>Lundi</th>
8       <th>Mardi</th>
```

## CalendarShowWeek

CalendarShowWeek est une requête GET qui permet de demander à l'API de nous renvoyer le fichier calendarWeekView.html qui correspond à la fenêtre affichant le calendrier dans sa version "week". Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/calendar/show/week.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

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

- Method:** GET
- URL:** {{LH}}calendar/show/week
- Status:** 200 OK
- Time:** 14 ms
- Size:** 10.99 KB
- Format:** HTML

The response body contains the following HTML code:

```
1 <div class="frame">
2   <table align="center" cellpadding="21" cellspacing="21" class="calendarWeek">
3
4     <caption align="top">
5
6     </caption>
7
8     <thead>
9       <tr>
10        <th class="secondTh"></th>
11        <th class="secondTh" id="WeekMon">Lundi</th>
```

## CalendarShowDay

CalendarShowDay est une requête GET qui permet de demander à l'API de nous renvoyer le fichier calendarDayView.html qui correspond à la fenêtre affichant le calendrier dans sa version "Day". Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/calendar/show/week.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

GET `{{LH}}calendar/show/day` Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 18 ms 3.44 KB Save Response

Pretty Raw Preview Visualize HTML

```
1 <div class="frame">
2   <table align="center" cellpadding="21" cellspacing="21" class="calendarDay">
3
4     <caption align="top">
5
6     </caption>
7
8     <thead>
9       <tr>
10        <th class="secondTh"></th>
```

## Login

Login est une requête GET qui permet de demander à l'API de nous renvoyer le fichier loginView.html qui correspond à la fenêtre de connexion de l'application. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/login.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

GET `{{LH}}`login Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 9 ms 1.39 KB Save Response

Pretty Raw Preview Visualize HTML Copy Search

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <title>DunPepene's Login</title>
6   <link href="/assets/croissant.png" rel="icon" type="image/png" />
7   <link href="../css/login.css" rel="stylesheet">
8   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
  display=swap" rel="stylesheet">
9 </head>
10
11 <body>
```

## Register

Register est une requête GET qui permet de demander à l'API de nous renvoyer le fichier registerView.html qui correspond à la fenêtre d'inscription de l'application. Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/register.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 200).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

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

- Method:** GET
- URL:** {{LH}}register
- Status:** 200 OK
- Time:** 17 ms
- Size:** 1.74 KB
- Response Format:** HTML

The response body contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <title>DunPepene's Register</title>
6   <link href="/assets/croissant.png" rel="icon" type="image/png" />
7   <link href="../css/register.css" rel="stylesheet">
8   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
  display=swap" rel="stylesheet">
9 </head>
10
11 <body>
```

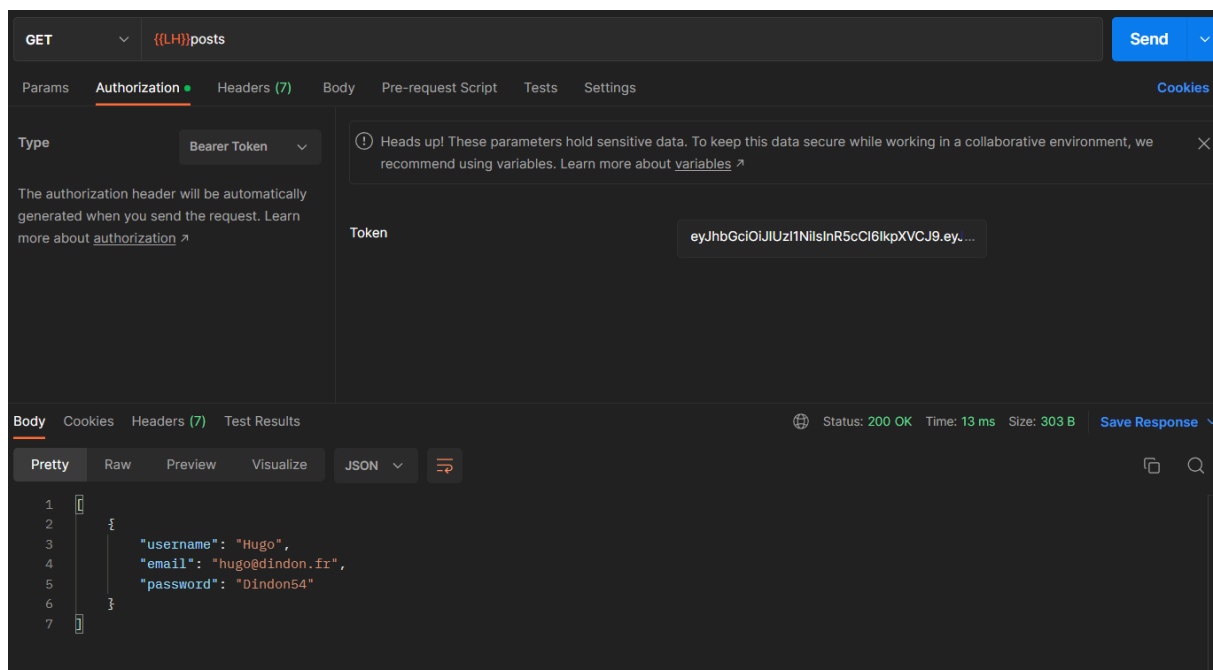
# Specials

## Posts

Posts est une requête GET qui permet de demander à l'API de nous renvoyer l'utilisateur connecté à partir d'un token JWT.

Le format de la requête est le suivant :

- La requête se formule comme cela : localhost:3030/posts.
- La requête requiert un bearer token.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code Json représentant les données liées à un utilisateur si la requête à abouti sans problème (status : 200).
  - Une erreur FORBIDDEN indiquant que le token est invalide (status : 403).
  - Une erreur UNAUTHORIZED indiquant que le token est n'est pas donné (status : 401).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple (401) ou bien de donner un token invalide (403)





401 est une requête GET qui permet de demander à l'API de nous renvoyer le fichier `error401View.html` qui correspond à la fenêtre d'erreur d'authentification. Le format de la requête est le suivant :

- La requête se formule comme cela : `localhost:3030/401`.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 401).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

The screenshot shows a web browser's developer tools interface. At the top, the request method is 'GET' and the URL is '{{LH}}401'. A 'Send' button is visible. Below the request bar, there are tabs for 'Params', 'Auth', 'Headers (6)', 'Body', 'Pre-req.', 'Tests', 'Settings', and 'Cookies'. The 'Body' tab is selected, showing the response content. The response status is '401 Unauthorized' with a response time of '24 ms' and a size of '922 B'. The response body is displayed in 'HTML' format, showing the following code:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <title>Error 401</title>
7    <link href="/assets/croissant.png" rel="icon" type="image/png" />
8    <link href="../css/error.css" rel="stylesheet">
9    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
      display=swap" rel="stylesheet">
10 </head>
11

```

500 est une requête GET qui permet de demander à l'API de nous renvoyer le fichier `error500View.html` qui correspond à la fenêtre d'erreur serveur. Le format de la requête est le suivant :

- La requête se formule comme cela : `localhost:3030/500`.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 500).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

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

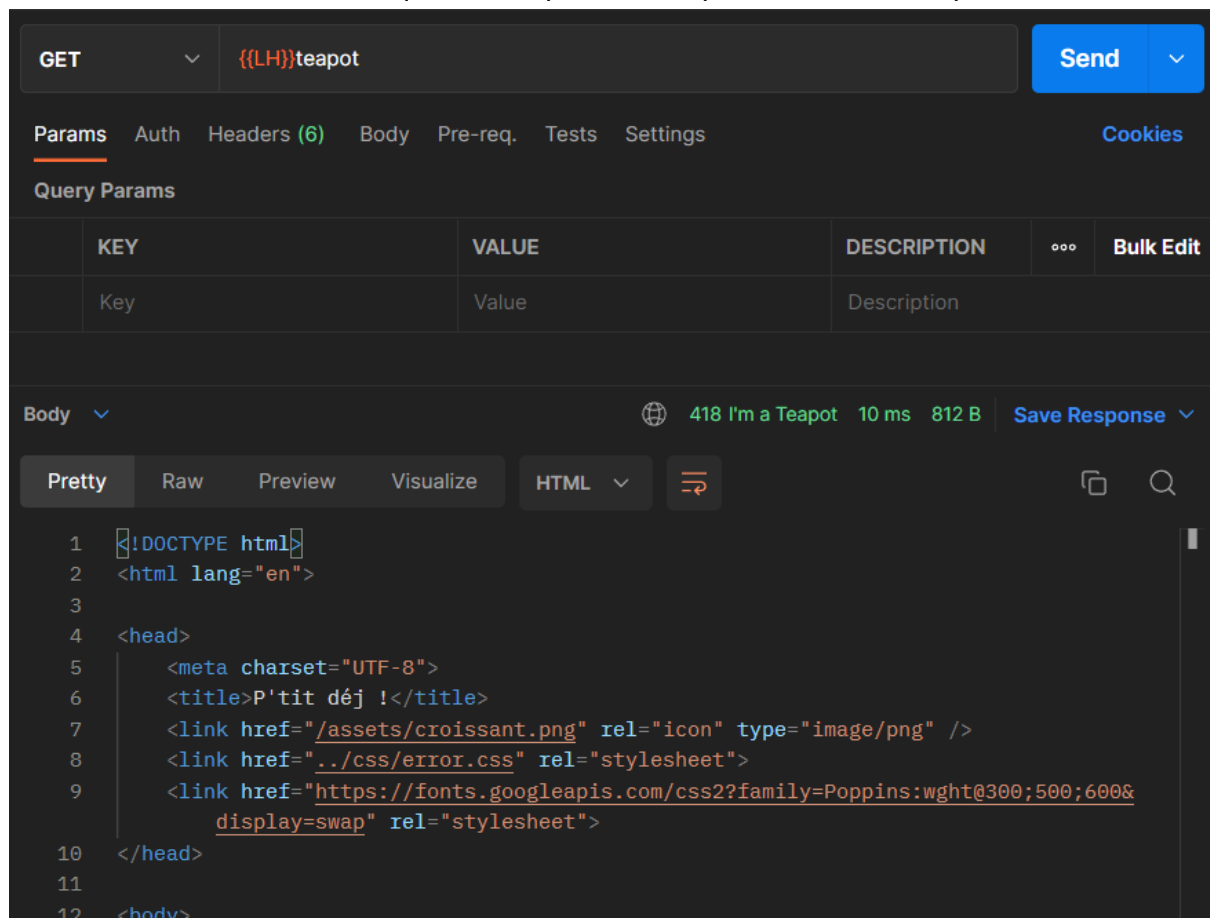
- Method:** GET
- URL:** `{{LH}}500`
- Status:** 500 Internal Server Error
- Time:** 14 ms
- Size:** 663 B
- Response Body (HTML):**

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>Error 500</title>
7   <link href="/assets/croissant.png" rel="icon" type="image/png" />
8   <link href="../css/error.css" rel="stylesheet">
9 </head>
10
```

## Teapot

Teapot est une requête GET qui permet de demander à l'API de nous renvoyer le fichier `teapotView.html` qui correspond à la fenêtre d'erreur serveur. Le format de la requête est le suivant :

- La requête se formule comme cela : `localhost:3030/teapot`.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 418).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.



GET `{{LH}}teapot` Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 418 I'm a Teapot 10 ms 812 B Save Response

Pretty Raw Preview Visualize HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>P'tit déj !</title>
7   <link href="/assets/croissant.png" rel="icon" type="image/png" />
8   <link href="../css/error.css" rel="stylesheet">
9   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&
    display=swap" rel="stylesheet">
10 </head>
11
12 <body>
```

404 est une requête GET qui permet de demander à l'API de nous renvoyer le fichier `teapotView.html` qui correspond à un easter egg. Le format de la requête est le suivant :

- La requête se formule avec n'importe quel "localhost:3030/XXX" possible.
- La requête ne requiert aucun paramètre.
- L'opération retourne plusieurs réponses selon l'aboutissement de la requête :
  - Le code html de la page sous format texte si la requête à abouti sans problème (status : 404).
  - Une erreur ENOENT indiquant que le fichier est introuvable (status : 404).
- Il suffit d'exécuter la requête sans paramètres pour avoir un exemple.

The screenshot shows a web browser's developer tools interface. At the top, the 'Specials' tab is selected, showing a '404' status. Below this, the 'GET' method is selected, and the URL is set to 'localhost:3030/404'. A 'Send' button is visible. The 'Params' tab is active, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table is empty. The 'Body' tab is selected, showing the response body in 'Pretty' format. The response is an HTML document with a 404 status, a title 'Error 404', and links to assets and stylesheets.

Specials / 404

GET {{LH}}404 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 404 Not Found 20 ms 651 B Save Response

Pretty Raw Preview Visualize HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>Error 404</title>
7   <link href="/assets/croissant.png" rel="icon" type="image/png" />
8   <link href="../css/error.css" rel="stylesheet">
9 </head>
10
```