

FRENCH↗

FRENCH

1. [CONFLUENCE] DOC POUR IMPORTATIONS

2. lien

3. Translating...

4. SWAGGER

5. Points de terminaison API

6. Liens d'ancrage ☐☐

6.1. Extrait de contenu de liens d'ancrage

7. Translating...

8. Tous les blocs

9. CATÉGORIE POUR GRILLES DE LIENS

9.1. Document de catégorie

9.1.1. JUSTE UN DOCUMENT SIMPLE

9.2. PREMIER DOCUMENT

9.2.1. 1.1 DOCUMENT

9.2.2. 1.2 DOCUMENT

9.2.3. 1.3 Document

9.2.4. DOCUMENT CACHÉ

9.3. DOCUMENT CACHÉ AVEC ENFANT

9.3.1. PREMIER DOCUMENT CACHÉ

9.4. DEUXIÈME DOCUMENT

9.4.1. 2.1 DOCUMENT

9.4.2. 2.2 DOCUMENT

9.4.3. 2.3 DOCUMENT

9.4.4. DOCUMENT CACHÉ

9.5. TROISIÈME DOCUMENT

9.5.1. 3.1 DOCUMENT

9.5.2. 3.2 DOCUMENT

9.5.3. 3.3 DOCUMENT

9.5.4. DOCUMENT CACHÉ

9.6. Sans titre

10. CODE DRAWERS (cloné)

11. Tiroirs de code

12. BLOCS INDIVIDUELS

12.1. Titre 1

12.2. Titre 2

12.3. Titre 3

13. Cartes de grilles de liens

14. Grilles de liens normales

15. QUESTIONS ET RÉPONSES

16. Quelques blocs

17. TEX

18. Exemple d'API

18.1. Lister tous les éléments.

18.2. Ajouter un élément.

1. [CONFLUENCE] DOC POUR IMPORTATIONS

Bonjour, ceci est un document qui devrait notifier l'utilisateur correctement.

Bonjour, ceci est un document qui devrait notifier le document [CONFLUENCE] DOCUMENT POUR MENTION ↗ correctement.

Bonjour, ceci est un document qui devrait contenir un hyperlien vers cette page ↗.

Ceci est un LIEN ↗

Ceci est un autre LIEN ↗

2. lien

Liens d'ancrage ☐☐

3. Translating...

4. SWAGGER

bla bla

Get Cakes eeee

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

Response

BODY PARAMETERS

id

String

required

ID du gâteau à obtenir

Parameter name

Integer

required

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");

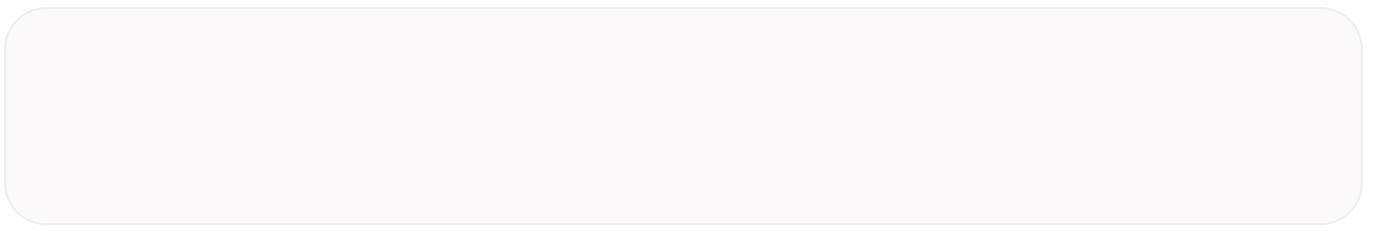
var raw = JSON.stringify({
  "id": "string"
});

var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};

fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

5. Points de terminaison API



Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


6. Liens d'ancrage □□

Titre 1

✓ Titre extensible 1

Contenu du titre extensible 1

Titre 2

✓ Titre extensible 2

Contenu du titre extensible 2

Titre 3

✓ Titre extensible 3

Contenu du titre extensible 3

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

Titre 1	Titre 2	Titre 3
Titre 1	Titre 2	Titre 3

Titre 1

Titre 2

Titre 3

▼ Titre extensible 1

Contenu

▼ Titre extensible 2

Contenu

▼ Titre extensible 3

Contenu

Titre 1

Titre 2

Titre 3

▼ Titre exp 1

Contenu

▼ Titre exp 2

Contenu

▼ Titre exp 3

Contenu

Titre 1

Titre 2

Titre 3

▼ Titre extensible 1

Contenu

▼ Titre extensible 2

Contenu

▼ Titre extensible 3

Contenu

Titre 1

2 Titre 2

Titre 2

3 Titre 3

Titre 3

4 ∨ Titre extensible 1

Contenu

5 ∨ Titre extensible 2

Contenu

6 ∨ Titre extensible 3

Contenu

Get Cakes

Obtenir un gâteau par son ID

GET<https://api.cakes.com>

Request

BODY PARAMETERS

id String **required**

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Nom du gâteau",
}
```

✓ Titre extensible 1

Contenu

✓ Titre extensible 2

Contenu

✓ Titre extensible 3

Contenu

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

Paste an image link or upload your image

Image URL

Add Image

/

6.1. Extrait de contenu de liens d'ancrage

Titre 1

✓ Titre extensible 1

Contenu du titre extensible 1

Titre 2

✓ Titre extensible 2

Contenu du titre extensible 2

Titre 3

✓ Titre extensible 3

Contenu du titre extensible 3

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Nom du gâteau",
}
```

Titre 1	Titre 2	Titre 3
Titre 1	Titre 2	Titre 3

Titre 1

Titre 2

Titre 3

▼ Titre extensible 1

Contenu

▼ Titre extensible 2

Contenu

▼ Titre extensible 3

Contenu

En-tête 1

En-tête 2

En-tête 3

▼ En-tête extensible 1

Contenu

▼ En-tête extensible 2

Contenu

▼ En-tête extensible 3

Contenu

En-tête 1

En-tête 2

En-tête 3

▼ En-tête extensible 1

Contenu

▼ En-tête extensible 2

Contenu

▼ En-tête extensible 3

Contenu

En-tête 1

2 En-tête 2

En-tête 2

3 En-tête 3

En-tête 3

4 ✓ En-tête extensible 1

Contenu

5 ✓ En-tête extensible 2

Contenu

6 ✓ En-tête extensible 3

Contenu

Get Cakes

Obtenir un gâteau par son ID

GET<https://api.cakes.com>

Request

BODY PARAMETERS

id String **required**

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Nom du gâteau",
}
```

✓ Titre extensible 1

Contenu

✓ Titre extensible 2

Contenu

✓ Titre extensible 3

Contenu

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Nom du gâteau",
}
```


7. Translating...

8. Tous les blocs

Titre 1 BOOBOO

▼

Titre extensible 1

Contenu du titre extensible 1

Titre 2

▼

Titre extensible 2

Contenu du titre extensible 2

Titre 3

▼

Titre extensible 3

Contenu du titre extensible 3

- ☐ Liste de contrôle
- ☐ Liste de contrôle 2
- ☐ Liste de contrôle 3

TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU

BOUTON

Encadré

En cours

MINI TÂCHE 1

Tests

MINI TÂCHE 2

Terminé

MINI TÂCHE 3



GRILLE DE LIENS 1



GRILLE DE LIENS 2



GRILLE DE LIENS 3

- 1

Workflow 1

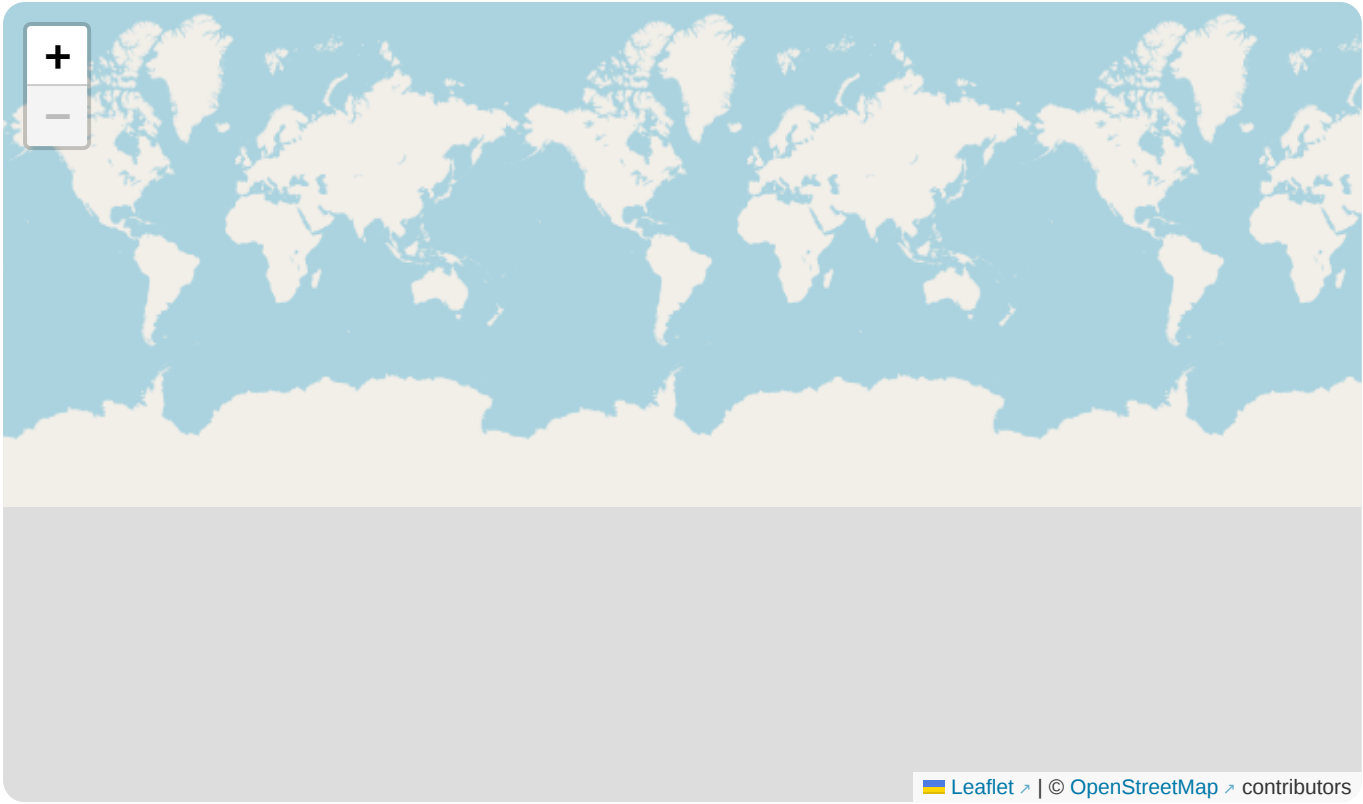
Contenu du workflow 1
- 2

Workflow 2

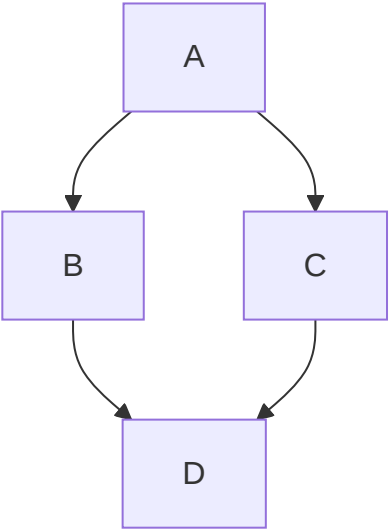
Contenu du workflow 2

Contenu de l'onglet 1

- Liste à puces 1
 - Liste à puces 2
 - Liste à puces 3
- 1. Liste numérotée 1
 - 1. Liste numérotée 2
 - 1. Liste numérotée 3



 Leaflet [↗](#) | ©  OpenStreetMap [↗](#) contributors



$$\int_0^i n f t y x^2 d x$$

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

1.0.7

OAS 2.0

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, [#swagger](https://github.com/swagger-api/swagger-ui). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)[Contact the developer](#)[Apache 2.0](#)[Find out more about Swagger](#)

Schemes

HTTPS

Authorize

**pet** Everything about your Pets[Find out more](#)

POST

/pet/{petId}/uploadImage uploads an image

Parameters

Try it out

Name

Description

petId * required

ID of pet to update

integer(\$int64)
(path)

petId

additionalMetadata

Additional data to pass to server

string
(formData)

additionalMetadata

file

file to upload

file
(formData)

Choose File No file chosen

Responses

Response content type

application/json

Code	Description
200	successful operation
Example Value Model <pre>{ "code": 0, "type": "string", "message": "string" }</pre>	

POST

/pet

Add a new pet to the store

Parameters

Try it out

Name	Description
body * required object (body)	Pet object that needs to be added to the store Example Value Model <pre>{ "id": 0, "category": { "id": 0, "name": "string" }, "name": "doggie", "photoUrls": ["string"], "tags": [{ "id": 0, "name": "string" }], "status": "available" }</pre> Parameter content type application/json

Responses

Response content type

application/json



Code

Description

405

Invalid input

PUT

/pet Update an existing pet



Parameters

Try it out

Name

Description

body * required

Pet object that needs to be added to the store

object
(body)

Example Value | Model

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
400	Invalid ID supplied
404	Pet not found
405	Validation exception

GET **/pet/findByStatus** Finds Pets by status



Multiple status values can be provided with comma separated strings

Parameters

Try it out

Name	Description
status * required array<string> <i>(query)</i>	Status values that need to be considered for filter <i>Available values : available, pending, sold</i>

available
pending
sold

▼ ▼ ▼

Responses

Response content type

application/json



Code	Description
200	successful operation

Example Value | Model

```
[
  {
    "id": 0,
    "category": {
      "id": 0,
      "name": "string"
    },
  },
]
```

Code**Description**

```
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
```

400

Invalid status value

GET

/pet/findByTags Finds Pets by tags

Warning: Deprecated

Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

Parameters

Try it out

Name**Description**

tags * required Tags to filter by
array<string>
(query)

Responses

Response content type

application/json

**Code****Description**

200

successful operation

Example Value | Model

```
[
  {
    "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
]
```

400

Invalid tag value

GET

/pet/{petId} Find pet by ID



Returns a single pet

Parameters

Try it out

Name

Description

petId * required ID of pet to return
integer(\$int64)
(path)

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value | Model

```
{
  "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
"status": "available"
}
```

- 400 Invalid ID supplied
- 404 Pet not found

POST

/pet/{petId} Updates a pet in the store with form data



Parameters

Try it out

Name Description

petId * required ID of pet that needs to be updated

integer(\$int64)
(path)

petId

name Updated name of the pet

string
(formData)

name

status Updated status of the pet

string
(formData)

status

Responses

Response content type

application/json



Code	Description
405	Invalid input

DELETE **/pet/{petId}** Deletes a pet



Parameters

Try it out

Name	Description
api_key string (header)	api_key
petId * required integer(\$int64) (path)	petId

Responses

Response content type

application/json



Code	Description
400	Invalid ID supplied
404	Pet not found

store Access to Petstore orders



GET **/store/inventory** Returns pet inventories by status



Returns a map of status codes to quantities

Parameters

Try it out

No parameters

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value | Model

```
{
  "additionalProp1": 0,
  "additionalProp2": 0,
  "additionalProp3": 0
}
```

POST

/store/order Place an order for a pet



Parameters

Try it out

Name

Description

body * required

order placed for purchasing the pet

object
(body)

Example Value | Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:45:48.350Z",
  "status": "placed",
  "complete": true
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0, "quantity": 0, "shipDate": "2026-06-05T08:45:48.425Z", "status": "placed", "complete": true }</pre>	
400	Invalid Order

GET

/store/order/{orderId}

Find purchase order by ID

^

For valid response try integer IDs with value ≥ 1 and ≤ 10 . Other values will generated exceptions

Parameters

Try it out

Name	Description
orderId * required integer(\$int64) (path)	ID of pet that needs to be fetched
orderId	

Responses

Response content type

application/json

▼

Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0,</pre>	

Code

Description

```
"quantity": 0,  
"shipDate": "2026-06-05T08:45:48.431Z",  
"status": "placed",  
"complete": true  
}
```

400

Invalid ID supplied

404

Order not found

DELETE

/store/order/{orderId} Delete purchase order by ID

For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors

Parameters

Try it out

Name

Description

orderId * required
integer(\$int64)
(path)

ID of the order that needs to be deleted

Responses

Response content type

application/json



Code

Description

400

Invalid ID supplied

404

Order not found

user Operations about user

Find out more about our store ↗ ^

POST

/user/createWithList Creates list of users with given input array

Parameters

Try it out

Name

Description

body * required

array<object>
(body)

List of user object

Example Value

Model

```
[
  {
    "id": 0,
    "username": "string",
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "password": "string",
    "phone": "string",
    "userStatus": 0
  }
]
```

Parameter content type

application/json

Responses

Response content type

application/json

Code	Description
default	successful operation

GET

/user/{username}

Get user by user name

Parameters

Try it out

Name

Description

username * required

string
(path)

The name that needs to be fetched. Use user1 for testing.

username

Responses

Response content type

application/json

Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }</pre>	
400	Invalid username supplied
404	User not found

PUT

/user/{username} Updated user



This can only be done by the logged in user.

Parameters

Try it out

Name	Description
username * required string (path)	name that need to be updated
username	
body * required object (body)	Updated user object
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string",</pre>	

Name

Description

```
"email": "string",
"password": "string",
"phone": "string",
"userStatus": 0
}
```

Parameter content type

application/json

Responses

Response content type

application/json

Code

Description

400

Invalid user supplied

404

User not found

DELETE

/user/{username}

Delete user

This can only be done by the logged in user.

Parameters

Try it out

Name

Description

username * required

The name that needs to be deleted

string

username

(path)

Responses

Response content type

application/json

Code	Description
------	-------------

400	Invalid username supplied
-----	---------------------------

404	User not found
-----	----------------

GET

/user/login Logs user into the system



Parameters

Try it out

Name	Description
------	-------------

username * required

The user name for login

string
(query)

username

password * required

The password for login in clear text

string
(query)

password

Responses

Response content type

application/json



Code	Description
------	-------------

200	successful operation
-----	----------------------

Example Value | Model

"string"

Headers:

Name	Description	Type
------	-------------	------

X-Expires-After	date in UTC when token expires	string
-----------------	--------------------------------	--------

X-Rate-Limit	calls per hour allowed by the user	integer
--------------	------------------------------------	---------

400	Invalid username/password supplied
-----	------------------------------------

GET**/user/logout** Logs out current logged in user session**Parameters****Try it out**

No parameters

Responses

Response content type

application/json

**Code****Description**

default

successful operation

POST**/user/createWithArray** Creates list of users with given input array**Parameters****Try it out****Name****Description****body** * required

List of user object

array<object>

(body)

Example Value | Model

```
[
  {
    "id": 0,
    "username": "string",
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "password": "string",
    "phone": "string",
    "userStatus": 0
  }
]
```

Parameter content type

application/json

**Responses**

Response content type

application/json



Code

Description

default

successful operation

POST

/user Create user



This can only be done by the logged in user.

Parameters

Try it out

Name

Description

body * required

Created user object

object
(body)

Example Value | Model

```
{
  "id": 0,
  "username": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "password": "string",
  "phone": "string",
  "userStatus": 0
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code

Description

default

successful operation

Models



ApiResponse >

Category >

Pet >

Tag >

Order >

User >



```
1 {  
2   status  
3   people  
4 }
```



Variables

Headers



+ GraphQL ↗

{}

sadasdas

Added

Contenu 1

Fixed

Contenu 2

Added

Added

9. CATÉGORIE POUR GRILLES DE LIENS

9.1. Document de catégorie

9.1.1. JUSTE UN DOCUMENT SIMPLE

9.2. PREMIER DOCUMENT

C'EST LE CONTENU DU PREMIER DOCUMENT.

9.2.1. 1.1 DOCUMENT

9.2.2. 1.2 DOCUMENT

9.2.3. 1.3 Document

9.2.4. DOCUMENT CACHÉ

9.3. DOCUMENT CACHÉ AVEC ENFANT

9.3.1. PREMIER DOCUMENT CACHÉ

9.4. DEUXIÈME DOCUMENT

VOICI LE CONTENU DU DEUXIÈME DOCUMENT.

9.4.2. 2.2 DOCUMENT

9.4.4. DOCUMENT CACHÉ

9.5. TROISIÈME DOCUMENT

C'EST LE CONTENU DU TROISIÈME DOCUMENT.

9.5.3. 3.3 DOCUMENT

9.5.4. DOCUMENT CACHÉ

9.6. Sans titre

title:

slug: liPn-

createdAt: Wed Dec 10 2025 13:15:20 GMT+0000 (Coordinated Universal Time)

updatedAt: Sun Dec 14 2025 09:05:19 GMT+0000 (Coordinated Universal Time)

10. CODE DRAWERS (cloné)

Titre 1

▼

Titre extensible 1

Contenu du titre extensible 1

Titre 2

▼

Titre extensible 2

Contenu du titre extensible 2

Titre 3

▼

Titre extensible 3

Contenu du titre extensible 3

- ☐ Liste de vérification
- ☐ Liste de vérification 2
- ☐ Liste de vérification 3

TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU

BOUTON

Encadré

En cours

MINI TÂCHE 1

Test

MINI TÂCHE 2

Terminé

MINI TÂCHE 3



GRILLE DE LIENS 1



GRILLE DE LIENS 2



GRILLE DE LIENS 3

- 1

Flux de travail 1

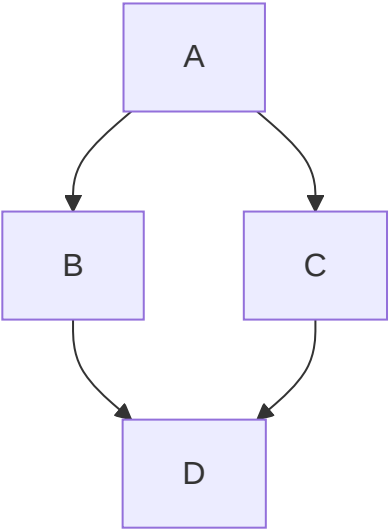
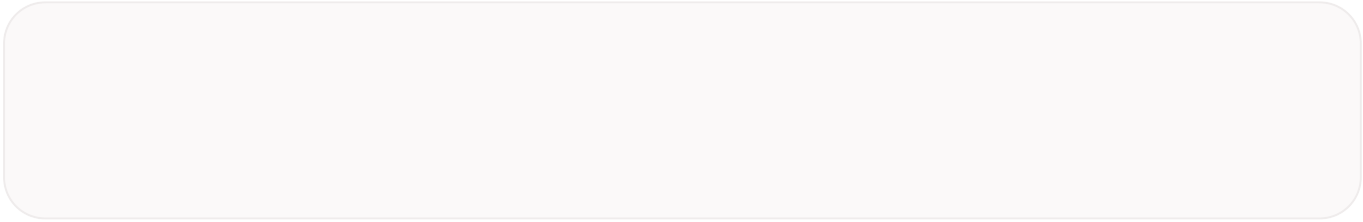
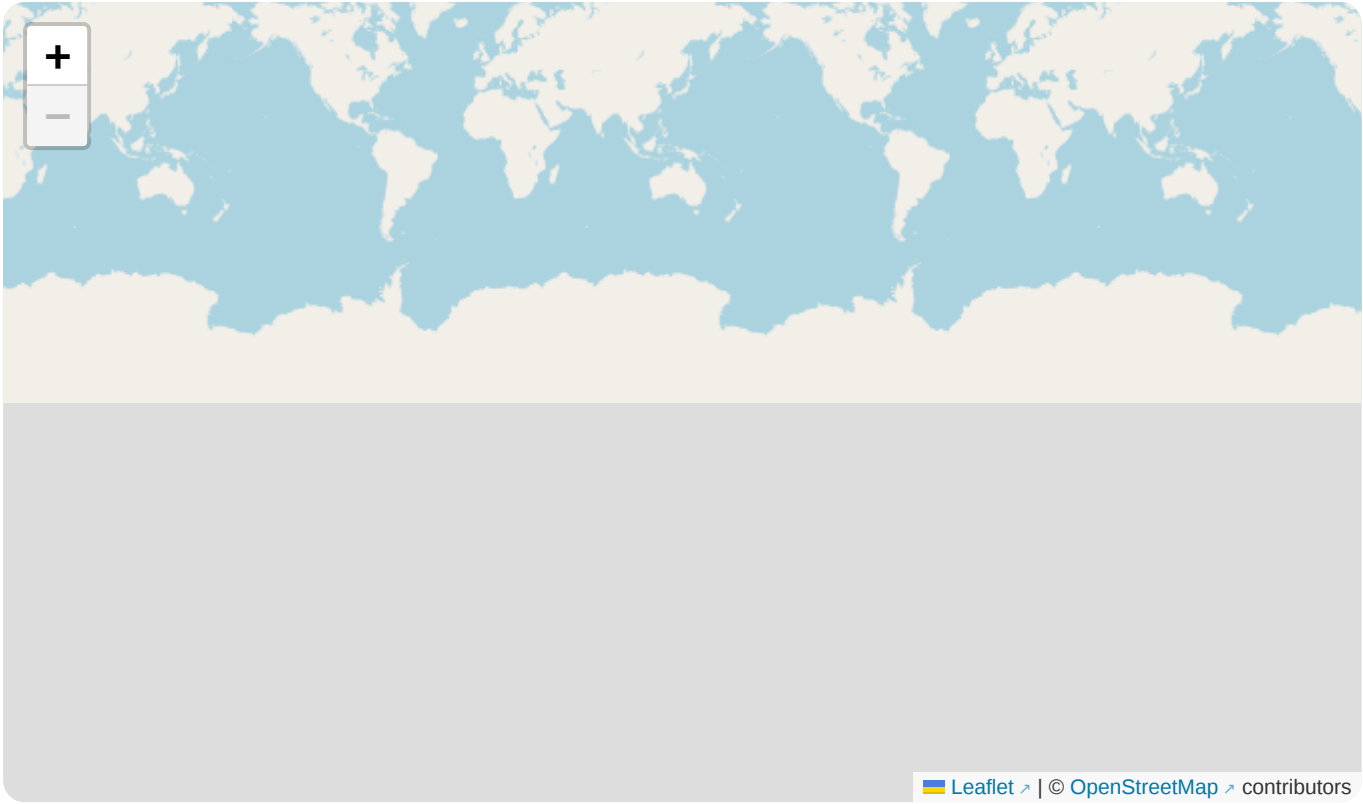
Contenu du flux de travail 1
- 2

Flux de travail 2

Contenu du flux de travail 2

Contenu de l'onglet 1

- Liste à puces 1
 - Liste à puces 2
 - Liste à puces 3
- Liste numérotée 1
 - Liste numérotée 2
 - Liste numérotée 3



$$\int_0^i n f t y x^2 d x$$

Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "String"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

1.0.7

OAS 2.0

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, [#swagger](https://github.com/swagger-api/swagger-ui). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)[Contact the developer](#)[Apache 2.0](#)[Find out more about Swagger](#)

Schemes

HTTPS



Authorize

**pet** Everything about your Pets[Find out more](#)

POST

/pet/{petId}/uploadImage uploads an image

Parameters

Try it out

Name

Description

petId * required

ID of pet to update

integer(\$int64)
(path)

petId

additionalMetadata

Additional data to pass to server

string
(formData)

additionalMetadata

file

file to upload

file
(formData)

Choose File No file chosen

Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "code": 0, "type": "string", "message": "string" }</pre>	

POST

/pet

Add a new pet to the store

Parameters

Try it out

Name	Description
body * required object (body)	Pet object that needs to be added to the store Example Value Model <pre>{ "id": 0, "category": { "id": 0, "name": "string" }, "name": "doggie", "photoUrls": ["string"], "tags": [{ "id": 0, "name": "string" }], "status": "available" }</pre> Parameter content type application/json

Responses

Response content type

application/json



Code

Description

405

Invalid input

PUT

/pet Update an existing pet



Parameters

Try it out

Name

Description

body * required

Pet object that needs to be added to the store

object
(body)

Example Value | Model

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
400	Invalid ID supplied
404	Pet not found
405	Validation exception

GET

/pet/findByStatus Finds Pets by status



Multiple status values can be provided with comma separated strings

Parameters

Try it out

Name	Description
status * required array<string> <i>(query)</i>	Status values that need to be considered for filter <i>Available values : available, pending, sold</i>



Responses

Response content type

application/json



Code	Description
200	successful operation

Example Value | Model

```
[
  {
    "id": 0,
    "category": {
      "id": 0,
      "name": "string"
    },
  },
]
```

Code**Description**

```
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
```

400

Invalid status value

GET

/pet/findByTags Finds Pets by tags

Warning: Deprecated

Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

Parameters

Try it out

Name**Description**

tags * required Tags to filter by
array<string>
(query)

Responses

Response content type

application/json

**Code****Description**

200

successful operation

Example Value | Model

```
[
  {
    "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
]
```

400

Invalid tag value

GET

/pet/{petId} Find pet by ID

Returns a single pet

Parameters

Try it out

Name

Description

petId * required

ID of pet to return

integer(\$int64)
(path)

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value | Model

```
{
  "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
"status": "available"
}
```

400

Invalid ID supplied

404

Pet not found

POST

/pet/{petId} Updates a pet in the store with form data

Parameters

Try it out

Name

Description

petId * required ID of pet that needs to be updatedinteger(\$int64)
(path)

name Updated name of the pet

string
(formData)

status Updated status of the pet

string
(formData)

Responses

Response content type

application/json



Code	Description
405	Invalid input

DELETE **/pet/{petId}** Deletes a pet



Parameters

Try it out

Name	Description
api_key string (header)	api_key
petId * required integer(\$int64) (path)	petId

Responses

Response content type

application/json



Code	Description
400	Invalid ID supplied
404	Pet not found

store Access to Petstore orders



GET **/store/inventory** Returns pet inventories by status



Returns a map of status codes to quantities

Parameters

Try it out

No parameters

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value

Model

```
{
  "additionalProp1": 0,
  "additionalProp2": 0,
  "additionalProp3": 0
}
```

POST

/store/order

Place an order for a pet



Parameters

Try it out

Name

Description

body * required

order placed for purchasing the pet

object
(body)

Example Value

Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:47:31.394Z",
  "status": "placed",
  "complete": true
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0, "quantity": 0, "shipDate": "2026-06-05T08:47:31.597Z", "status": "placed", "complete": true }</pre>	
400	Invalid Order

GET

/store/order/{orderId} Find purchase order by ID



For valid response try integer IDs with value ≥ 1 and ≤ 10 . Other values will generated exceptions

Parameters

Try it out

Name	Description
orderId * required integer(\$int64) (path)	ID of pet that needs to be fetched
orderId	

Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0,</pre>	

Code	Description
	<pre>"quantity": 0, "shipDate": "2026-06-05T08:47:31.608Z", "status": "placed", "complete": true }</pre>
400	Invalid ID supplied
404	Order not found

DELETE **/store/order/{orderId}** Delete purchase order by ID ^

For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors

Parameters Try it out

Name	Description
orderId * required integer(\$int64) (path)	ID of the order that needs to be deleted

Responses Response content type **application/json** v

Code	Description
400	Invalid ID supplied
404	Order not found

user Operations about user

Find out more about our store ↗ ^

POST **/user/createWithList** Creates list of users with given input array ^

Parameters Try it out

Name

Description

body * required

array<object>
(body)

List of user object

Example Value

Model

```
[
  {
    "id": 0,
    "username": "string",
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "password": "string",
    "phone": "string",
    "userStatus": 0
  }
]
```

Parameter content type

application/json

Responses

Response content type

application/json

Code	Description
default	successful operation

GET /user/{username} Get user by user name

Parameters

Try it out

Name

Description

username * required

string
(path)

The name that needs to be fetched. Use user1 for testing.

username

Responses

Response content type

application/json

Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }</pre>	
400	Invalid username supplied
404	User not found

PUT

/user/{username} Updated user



This can only be done by the logged in user.

Parameters

Try it out

Name	Description
username * required string (path)	name that need to be updated
username	
body * required object (body)	Updated user object
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string",</pre>	

Name

Description

```
"email": "string",  
"password": "string",  
"phone": "string",  
"userStatus": 0  
}
```

Parameter content type

application/json

Responses

Response content type

application/json

Code

Description

400

Invalid user supplied

404

User not found

DELETE

/user/{username}

Delete user

This can only be done by the logged in user.

Parameters

Try it out

Name

Description

username * required

The name that needs to be deleted

string

username

(path)

Responses

Response content type

application/json

Code	Description
------	-------------

400	Invalid username supplied
-----	---------------------------

404	User not found
-----	----------------

GET

/user/login Logs user into the system



Parameters

Try it out

Name	Description
------	-------------

username * required

The user name for login

string
(query)

username

password * required

The password for login in clear text

string
(query)

password

Responses

Response content type

application/json



Code	Description
------	-------------

200	successful operation
-----	----------------------

Example Value | Model

"string"

Headers:

Name	Description	Type
------	-------------	------

X-Expires-After	date in UTC when token expires	string
-----------------	--------------------------------	--------

X-Rate-Limit	calls per hour allowed by the user	integer
--------------	------------------------------------	---------

400	Invalid username/password supplied
-----	------------------------------------

GET**/user/logout** Logs out current logged in user session**Parameters**[Try it out](#)

No parameters

Responses

Response content type

application/json

**Code****Description**

default

successful operation

POST**/user/createWithArray** Creates list of users with given input array**Parameters**[Try it out](#)**Name****Description****body** * required
array<object>
(body)

List of user object

Example Value | Model

```
[
  {
    "id": 0,
    "username": "string",
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "password": "string",
    "phone": "string",
    "userStatus": 0
  }
]
```

Parameter content type

application/json

**Responses**

Response content type

application/json



Code

Description

default

successful operation

POST

/user Create user



This can only be done by the logged in user.

Parameters

Try it out

Name

Description

body * required

Created user object

object
(body)

Example Value | Model

```
{
  "id": 0,
  "username": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "password": "string",
  "phone": "string",
  "userStatus": 0
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code

Description

default

successful operation

Models



ApiResponse >

Category >

Pet >

Tag >

Order >

User >



```
1 {  
2   status,  
3   people  
4 }
```



Variables

Headers



+ GraphQL ↗

{}

Changelog Title

Added

Content 1

Fixed

Content 2

Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


11. Tiroirs de code

Titre 1

▼

Titre extensible 1

Contenu du titre extensible 1

Titre 2

▼

Titre extensible 2

Contenu du titre extensible 2

Titre 3

▼

Titre extensible 3

Contenu du titre extensible 3

- ☐ Liste de vérification
- ☐ Liste de vérification 2
- ☐ Liste de vérification 3

TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU

BOUTON

Encadré

Doing

MINI TÂCHE 1

Testing

MINI TÂCHE 2

Done

MINI TÂCHE 3



GRILLE DE LIENS 1



GRILLE DE LIENS 2



GRILLE DE LIENS 3

1

Flux de travail 1

Contenu du flux de travail 1

2

Flux de travail 2

Contenu du flux de travail 2

ONGLET 1

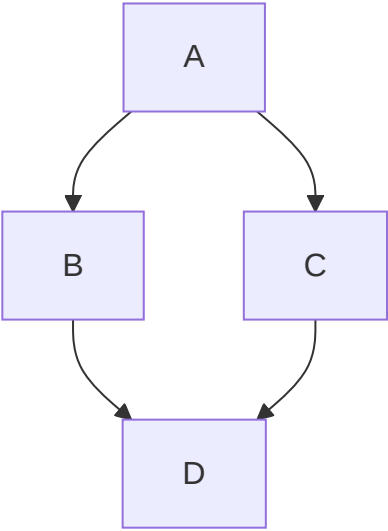
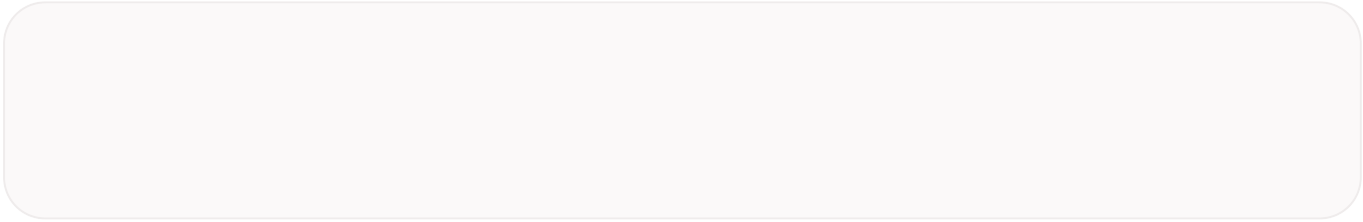
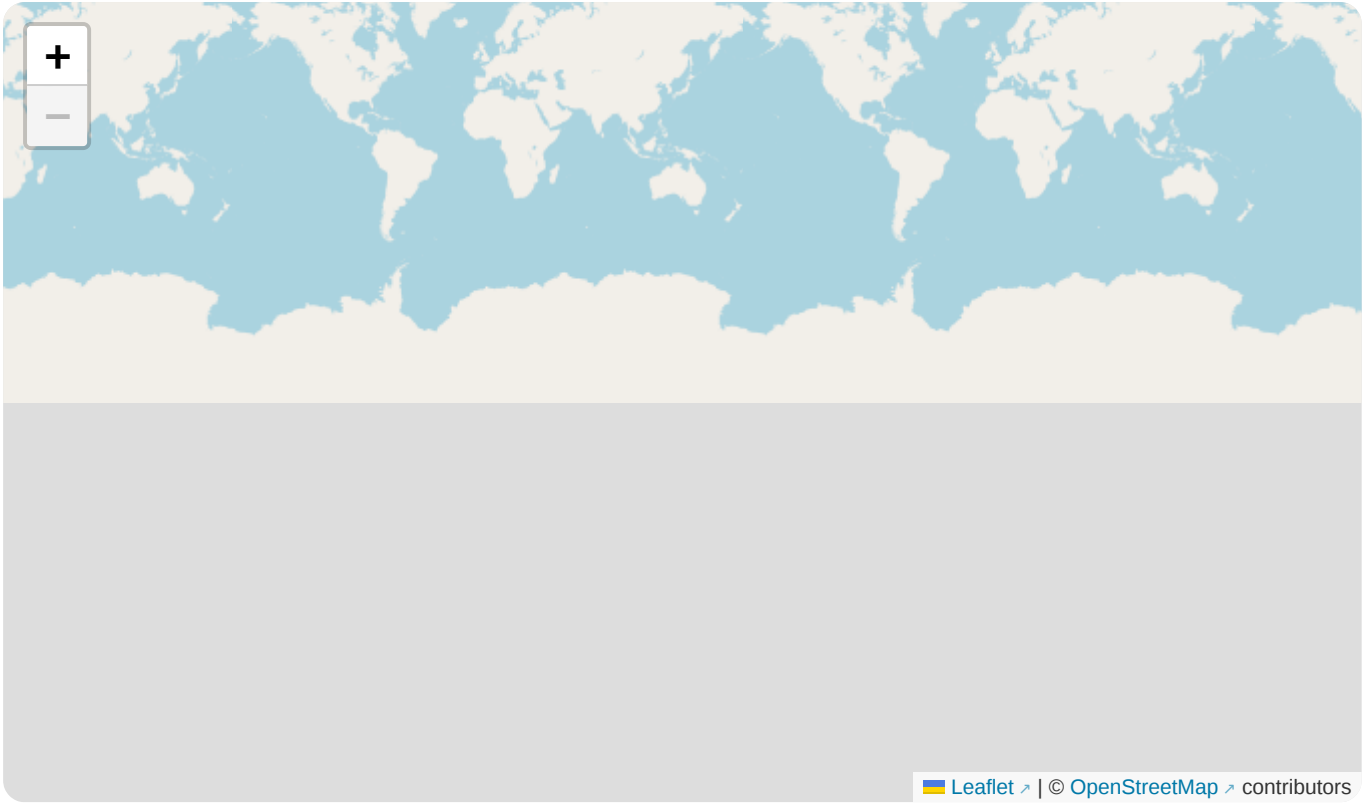
ONGLET 2

Nouvel onglet

Nouvel onglet

Contenu de l'onglet 1

- Liste à puces 1
 - Liste à puces 2
 - Liste à puces 3
- 1. Liste numérotée 1
 - 1. Liste numérotée 2
 - 1. Liste numérotée 3



$$\int_0^i n f t y x^2 d x$$

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("accept", "application/json");
myHeaders.append("content-type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

1.0.7

OAS 2.0

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, [#swagger](https://t.me/swagger). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)[Contact the developer](#)[Apache 2.0](#)[Find out more about Swagger](#)

Schemes

HTTPS



Authorize

**pet** Everything about your Pets[Find out more](#)

POST

/pet/{petId}/uploadImage uploads an image

Parameters

Try it out

Name

Description

petId * required

ID of pet to update

integer(\$int64)
(path)

petId

additionalMetadata

Additional data to pass to server

string
(formData)

additionalMetadata

file

file to upload

file
(formData)

Choose File No file chosen

Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "code": 0, "type": "string", "message": "string" }</pre>	

POST

/pet

Add a new pet to the store

Parameters

Try it out

Name	Description
body * required object (body)	Pet object that needs to be added to the store Example Value Model <pre>{ "id": 0, "category": { "id": 0, "name": "string" }, "name": "doggie", "photoUrls": ["string"], "tags": [{ "id": 0, "name": "string" }], "status": "available" }</pre> Parameter content type application/json

Responses

Response content type

application/json



Code

Description

405

Invalid input

PUT

/pet Update an existing pet



Parameters

Try it out

Name

Description

body * required

Pet object that needs to be added to the store

object
(body)

Example Value | Model

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
400	Invalid ID supplied
404	Pet not found
405	Validation exception

GET

/pet/findByStatus Finds Pets by status

Multiple status values can be provided with comma separated strings

Parameters

Try it out

Name	Description
------	-------------

status * required
array<string>
(query)

Status values that need to be considered for filter

Available values : available, pending, sold

available
 pending
 sold



Responses

Response content type

application/json



Code	Description
------	-------------

200	successful operation
-----	----------------------

Example Value | Model

```
[
  {
    "id": 0,
    "category": {
      "id": 0,
      "name": "string"
    },
  },
]
```

Code**Description**

```
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
```

400

Invalid status value

GET

/pet/findByTags Finds Pets by tags

Warning: Deprecated

Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

Parameters

Try it out

Name**Description**

tags * required Tags to filter by
array<string>
(query)

Responses

Response content type

application/json

**Code****Description**

200

successful operation

Example Value | Model

```
[
  {
    "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
]
```

400

Invalid tag value

GET

/pet/{petId} Find pet by ID



Returns a single pet

Parameters

Try it out

Name

Description

petId * required ID of pet to return
integer(\$int64)
(path)

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value | Model

```
{
  "id": 0,
```

Code

Description

```
"category": {
  "id": 0,
  "name": "string"
},
"name": "doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
"status": "available"
}
```

400

Invalid ID supplied

404

Pet not found

POST

/pet/{petId} Updates a pet in the store with form data

Parameters

Try it out

Name

Description

petId * required ID of pet that needs to be updatedinteger(\$int64)
(path)

name Updated name of the pet

string
(formData)

status Updated status of the pet

string
(formData)

Responses

Response content type

application/json



Code	Description
405	Invalid input

DELETE

/pet/{petId}
Deletes a pet

Parameters

Try it out

Name	Description
api_key string <i>(header)</i>	api_key
petId * required integer(\$int64) <i>(path)</i>	Pet id to delete petId

Responses

Response content type

application/json

Code	Description
400	Invalid ID supplied
404	Pet not found

store
Access to Petstore orders

GET

/store/inventory
Returns pet inventories by status

Returns a map of status codes to quantities

Parameters

Try it out

No parameters

Responses

Response content type

application/json



Code

Description

200

successful operation

Example Value | Model

```
{
  "additionalProp1": 0,
  "additionalProp2": 0,
  "additionalProp3": 0
}
```

POST

/store/order Place an order for a pet



Parameters

Try it out

Name

Description

body * required
object
(body)

order placed for purchasing the pet

Example Value | Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:47:31.411Z",
  "status": "placed",
  "complete": true
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0, "quantity": 0, "shipDate": "2026-06-05T08:47:31.733Z", "status": "placed", "complete": true }</pre>	
400	Invalid Order

GET

/store/order/{orderId}

Find purchase order by ID

^

For valid response try integer IDs with value ≥ 1 and ≤ 10 . Other values will generated exceptions

Parameters

Try it out

Name	Description
orderId * required integer(\$int64) (path)	ID of pet that needs to be fetched
	orderId

Responses

Response content type

application/json

▼

Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "petId": 0,</pre>	

Code	Description
	<pre>"quantity": 0, "shipDate": "2026-06-05T08:47:31.743Z", "status": "placed", "complete": true }</pre>
400	Invalid ID supplied
404	Order not found

DELETE **/store/order/{orderId}** Delete purchase order by ID

For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors

Parameters Try it out

Name	Description
orderId * required integer(\$int64) (path)	ID of the order that needs to be deleted

orderId

Responses Response content type application/json

Code	Description
400	Invalid ID supplied
404	Order not found

user Operations about user

Find out more about our store ↗ ^

POST **/user/createWithList** Creates list of users with given input array

Parameters Try it out

Name	Description
body * required array<object> (body)	List of user object
	Example Value Model <pre>[{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }]</pre>
	Parameter content type application/json

Responses

Response content type

application/json

Code	Description
default	successful operation

GET /user/{username} Get user by user name

Parameters

Try it out

Name	Description
username * required string (path)	The name that needs to be fetched. Use user1 for testing.

username

Responses

Response content type

application/json

Code	Description
200	successful operation
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }</pre>	
400	Invalid username supplied
404	User not found

PUT

/user/{username} Updated user



This can only be done by the logged in user.

Parameters

Try it out

Name	Description
username * required string (path)	name that need to be updated
username	
body * required object (body)	Updated user object
Example Value Model <pre>{ "id": 0, "username": "string", "firstName": "string", "lastName": "string",</pre>	

Name	Description
	<pre>"email": "string", "password": "string", "phone": "string", "userStatus": 0 }</pre>
Parameter content type	
application/json	

Responses

Response content type

application/json

Code	Description
400	Invalid user supplied
404	User not found

DELETE

/user/{username}

Delete user

This can only be done by the logged in user.

Parameters

Try it out

Name	Description
username * required string (path)	The name that needs to be deleted
	username

Responses

Response content type

application/json

Code	Description
------	-------------

400	Invalid username supplied
-----	---------------------------

404	User not found
-----	----------------

GET

/user/login Logs user into the system



Parameters

Try it out

Name	Description
------	-------------

username * required

The user name for login

string
(query)

username

password * required

The password for login in clear text

string
(query)

password

Responses

Response content type

application/json



Code	Description
------	-------------

200	successful operation
-----	----------------------

Example Value | Model

"string"

Headers:

Name	Description	Type
------	-------------	------

X-Expires-After	date in UTC when token expires	string
-----------------	--------------------------------	--------

X-Rate-Limit	calls per hour allowed by the user	integer
--------------	------------------------------------	---------

400	Invalid username/password supplied
-----	------------------------------------

GET**/user/logout** Logs out current logged in user session**Parameters****Try it out**

No parameters

Responses

Response content type

application/json

**Code****Description**

default

successful operation

POST**/user/createWithArray** Creates list of users with given input array**Parameters****Try it out****Name****Description****body** * required
array<object>
(body)

List of user object

Example Value | Model

```
[
  {
    "id": 0,
    "username": "string",
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "password": "string",
    "phone": "string",
    "userStatus": 0
  }
]
```

Parameter content type

application/json

**Responses**

Response content type

application/json



Code

Description

default

successful operation

POST

/user Create user



This can only be done by the logged in user.

Parameters

Try it out

Name

Description

body * required

Created user object

object
(body)

Example Value | Model

```
{
  "id": 0,
  "username": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "password": "string",
  "phone": "string",
  "userStatus": 0
}
```

Parameter content type

application/json



Responses

Response content type

application/json



Code

Description

default

successful operation

Models



ApiResponse >

Category >

Pet >

Tag >

Order >

User >



```
1 ▾ {  
2   status,  
3   people  
4 }
```



Variables

Headers



+ GraphQL ↗

{}

Titre du changelog

Added

Contenu 1

Fixed

Contenu 2

Get Cakes

Obtenir un gâteau par son ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID du gâteau à obtenir

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


Get Cakes

Get a cake by its ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");
```

```
var raw = JSON.stringify({
  "id": "string"
});
```

```
var requestOptions = {
  method: 'GET',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};
```

```
fetch("https://api.cakes.com", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```


12. BLOCS INDIVIDUELS

12.1. Titre 1

Titre 1

12.2. Titre 2

Titre 2

12.3. Titre 3

Titre 3

13. Cartes de grilles de liens

PREMIER DOCUMENT

DEUXIÈME DOCUMENT

TROISIÈME DOCUMENT

14. Grilles de liens normales

PREMIER DOCUMENT

DEUXIÈME DOCUMENT

TROISIÈME DOCUMENT

15. QUESTIONS ET RÉPONSES

Juste un document.

Cliquez pour modifier le libellé du bouton

Cliquez pour modifier le libellé du bouton

16. Quelques blocs

Titre 1

▼ Titre déroulant 1

Contenu du titre déroulant 1

Titre 2

▼ Titre déroulant 2

Contenu du titre déroulant 2

Titre 3

▼ Titre déroulant 3

Contenu du titre déroulant 3

- ☐ Liste de vérification
- ☐ Liste de vérification 2
- ☐ Liste de vérification 3

TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU
TABLEAU	TABLEAU	TABLEAU

BOUTON

Encadré

En cours

TÂCHE MINI 1

Testing

TÂCHE MINI 2

Done

TÂCHE MINI 3



GRILLE DE LIENS 1



GRILLE DE LIENS 2



GRILLE DE LIENS 3

- 1

Flux de travail 1

Contenu du flux de travail 1
- 2

Flux de travail 2

Contenu du flux de travail 2

Contenu de l'onglet 1

- Liste à puces 1
 - Liste à puces 2
 - Liste à puces 3
- 1. Liste numérotée 1
 - 1. Liste numérotée 2
 - 1. Liste numérotée 3

$$m_{\text{yeast}} = \left(\frac{m_{\text{sample,dry}}}{V_{\text{sample}}} \right) V_{\text{final}}$$

$$Y_{\text{fermenter}} = \frac{m_{\text{yeast}}}{m_{\text{molasses}}}$$

$$\eta = \begin{cases} 1, & \text{if } T \geq 30^\circ\text{C} \\ 0, & \text{if } T < 30^\circ\text{C} \end{cases}$$

$$c = \frac{m}{V} \text{ [g/L]}$$

$$\begin{aligned} m_{\text{yeast}} &= \left(\frac{m_{\text{sample,dry}}}{V_{\text{sample}}} \right) V_{\text{final}} \\ Y_{\text{fermenter}} &= \frac{m_{\text{yeast}}}{m_{\text{molasses}}} \\ \eta(T) &= \begin{cases} 1, & \text{if } 30^\circ\text{C} \leq T \leq 35^\circ\text{C} \\ 0.8, & \text{if } 25^\circ\text{C} \leq T < 30^\circ\text{C} \\ 0, & \text{if } T < 25^\circ\text{C} \end{cases} \\ c_{\text{yeast}} &= \frac{m_{\text{sample,dry}}}{V_{\text{sample}}} \text{ [g/L]} \end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} m_{\text{sample,dry}} \\ V_{\text{sample}} \\ V_{\text{final}} \\ m_{\text{molasses}} \end{bmatrix}, \quad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \left(\frac{x_1}{x_2} \right) x_3 \\ \frac{\left(\frac{x_1}{x_2} \right) x_3}{x_4} \end{bmatrix}$$
$$m_{\text{yeast}} = f_1(\mathbf{x}), \quad Y_{\text{fermenter}} = f_2(\mathbf{x})$$

$$m_{\text{yeast}} = \left(\frac{m_{\text{sample,dry}}}{V_{\text{sample}}} \right) V_{\text{final}}$$

$$\ln(m_{\text{yeast}}) = \ln(m_{\text{sample,dry}}) - \ln(V_{\text{sample}}) + \ln(V_{\text{final}})$$

$$\left(\frac{\sigma_{m_{\text{yeast}}}}{m_{\text{yeast}}} \right)^2 = \left(\frac{\sigma_{m_{\text{sample,dry}}}}{m_{\text{sample,dry}}} \right)^2 + \left(\frac{\sigma_{V_{\text{sample}}}}{V_{\text{sample}}} \right)^2 + \left(\frac{\sigma_{V_{\text{final}}}}{V_{\text{final}}} \right)^2$$

$$Y_{\text{fermenter}} = \frac{m_{\text{yeast}}}{m_{\text{molasses}}}$$

Given: $m_{\text{sample,dry}}, V_{\text{sample}}, V_{\text{final}}, m_{\text{molasses}}, T, t, t_f$

Concentration: $c_{\text{yeast}} = \frac{m_{\text{sample,dry}}}{V_{\text{sample}}} \text{ [g/L]}$

Total yeast: $m_{\text{yeast}} = c_{\text{yeast}} V_{\text{final}}$

Fermenter yield: $Y_{\text{fermenter}} = \frac{m_{\text{yeast}}}{m_{\text{molasses}}}$

Temperature factor: $\eta(T) = \begin{cases} 0, & T < 25^\circ\text{C} \\ 0.8 + 0.04 (T - 25^\circ\text{C}), & 25^\circ\text{C} \leq T < 30^\circ\text{C} \\ 1, & 30^\circ\text{C} \leq T \leq 35^\circ\text{C} \\ \exp(-0.2 (T - 35^\circ\text{C})), & T > 35^\circ\text{C} \end{cases}$

A simple growth profile: $m_{\text{yeast}}(t) = \frac{m_{\text{max}}}{1 + \exp(-k \eta(T) (t - t_0))}$

Net production (from 0 to t_f): $\Delta m_{\text{yeast}} = m_{\text{yeast}}(t_f) - m_{\text{yeast}}(0)$

Mass balance check: $\epsilon = \frac{\Delta m_{\text{yeast}}}{m_{\text{molasses}}} - Y_{\text{target}}$

Decision: pass if $|\epsilon| \leq 0.02$, fail otherwise

Uncertainty propagation (relative): $\left(\frac{\sigma_{m_{\text{yeast}}}}{m_{\text{yeast}}} \right)^2 = \left(\frac{\sigma_{m_{\text{sample,dry}}}}{m_{\text{sample,dry}}} \right)^2 + \left(\frac{\sigma_{V_{\text{sample}}}}{V_{\text{sample}}} \right)^2$

And for yield: $\left(\frac{\sigma_{Y_{\text{fermenter}}}}{Y_{\text{fermenter}}} \right)^2 = \left(\frac{\sigma_{m_{\text{yeast}}}}{m_{\text{yeast}}} \right)^2 + \left(\frac{\sigma_{m_{\text{molasses}}}}{m_{\text{molasses}}} \right)^2$

Inputs: $m_{\text{sample,dry}} = 1.25 \text{ g}$, $V_{\text{sample}} = 25 \text{ mL}$, $V_{\text{final}} = 18,000 \text{ L}$, m_{m}

Unit conversions: $V_{\text{sample}} = 25 \times 10^{-3} \text{ L}$, $m_{\text{molasses}} = 9.5 \times 10^6 \text{ g}$

Step 1 (conc.): $c_{\text{yeast}} = \frac{1.25}{25 \times 10^{-3}} = 50 \text{ g/L}$

Step 2 (total): $m_{\text{yeast}} = 50 \times 18,000 = 9.0 \times 10^5 \text{ g}$

Step 3 (yield): $Y_{\text{fermenter}} = \frac{9.0 \times 10^5}{9.5 \times 10^6} = 0.0947$

Constraint checks: $0 < Y_{\text{fermenter}} < 0.2$, $V_{\text{final}} > 0$, $V_{\text{sample}} > 0$

Vector form: $\mathbf{x} = \begin{bmatrix} m_{\text{sample,dry}} \\ V_{\text{sample}} \\ V_{\text{final}} \\ m_{\text{molasses}} \end{bmatrix}$, $\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \left(\frac{x_1}{x_2} \right) x_3 \\ \frac{\left(\frac{x_1}{x_2} \right) x_3}{x_4} \end{bmatrix}$

Outputs: $\begin{bmatrix} m_{\text{yeast}} \\ Y_{\text{fermenter}} \end{bmatrix} = \mathbf{g}(\mathbf{x})$

18. Exemple d'API

18.1. Lister tous les éléments.

GET

https://api.example.com/v1/item

Try it ▶

Optional server description, e.g. Main (production) server

Lists all the items.

Responses

▼ ● 200 OK

▼ item[]

> bad integer

> good integer

```
curl --request GET \  
  --url https://api.example.com/v1/item \  
  --header 'accept: application/json'
```

```
// OK
[
  {
    "bad": 0,
    "good": 1
  }
]
```

18.2. Ajouter un élément.

POST

https://api.example.com/v1/item

Try it ▶

Optional server description, e.g. Main (production) server

Credentials

HTTP Bearer Bearer

Enter http bearer value

Body Parameters

▼ body item

> bad integer

> good integer

Responses

● 200 OK

```
curl --request POST \  
  --url https://api.example.com/v1/item \  
  --header 'accept: application/json'
```

```
// OK
```