

SPANISH ↗

SPANISH

1. [CONFLUENCE] DOC FOR IMPORTS
2. enlace
3. Translating...
4. SWAGGER
5. Puntos de extremo de API
6. ENLACES ANCLA 
 - 6.1. Fragmento de contenido de enlaces ancla
7. Translating...
8. Todos los bloques
9. CATEGORÍA PARA REJILLAS DE ENLACES
 - 9.1. DOCUMENTO DE CATEGORÍA
 - 9.1.1. SOLO UN DOCUMENTO SIMPLE
 - 9.2. PRIMER DOCUMENTO
 - 9.2.1. 1.1 DOCUMENTO
 - 9.2.2. 1.2 DOCUMENT
 - 9.2.3. 1.3 DOCUMENTO
 - 9.2.4. DOCUMENTO OCULTO
 - 9.3. DOCUMENTO OCULTO CON HIJO
 - 9.3.1. Primer documento oculto
 - 9.4. Segundo documento
 - 9.4.1. 2.1 DOCUMENTO
 - 9.4.2. 2.2 DOCUMENTO
 - 9.4.3. 2.3 DOCUMENTO
 - 9.4.4. DOCUMENTO OCULTO
 - 9.5. TERCER DOCUMENTO
 - 9.5.1. 3.1 DOCUMENTO
 - 9.5.2. 3.2 DOCUMENTO
 - 9.5.3. 3.3 DOCUMENT

9.5.4. Documento oculto

9.6. Sin título

10. CODE DRAWERS (clonado)

11. CODE DRAWERS

12. BLOQUES INDIVIDUALES

12.1. Encabezado 1

12.2. Encabezado 2

12.3. Encabezado 3

13. ENLACE CUADRÍCULAS TARJETAS

14. ENLACE CUADRÍCULAS NORMAL

15. Preguntas y respuestas

16. ALGUNOS BLOQUES

17. TEX

18. API de muestra

18.1. Listar todos los ítems.

18.2. Agregar un elemento.

1. [CONFLUENCE] DOC FOR IMPORTS

Hola, este es un documento que debería notificar al usuario correctamente.

Hola, este es un documento que debería notificar al documento [CONFLUENCE] DOC FOR MENTION ↗ correctamente.

Hola, este es un documento que debería tener un hipervínculo a esta página ↗.

Este es un ENLACE ↗

Este es otro ENLACE ↗

2. enlace

ENLACES ANCLA []

3. Translating...

4. SWAGGER

bla bla

Get Cakes eeee

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

Response

BODY PARAMETERS

id

String

required

ID del pastel a obtener

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. Puntos de extremo de 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. ENLACES ANCLA

Encabezado 1

✓ Encabezado Expandible 1

Contenido del Encabezado Expandible 1

Encabezado 2

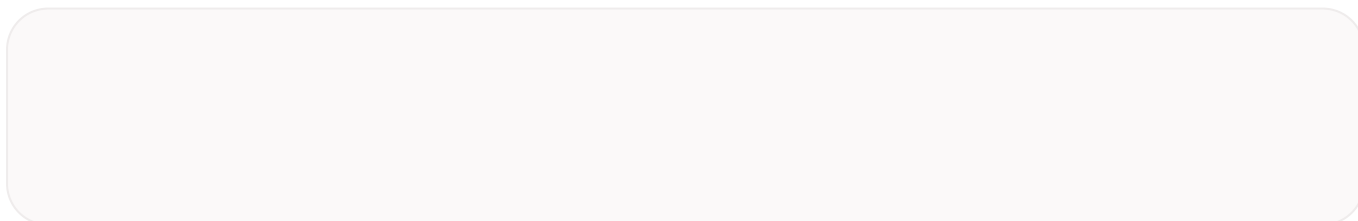
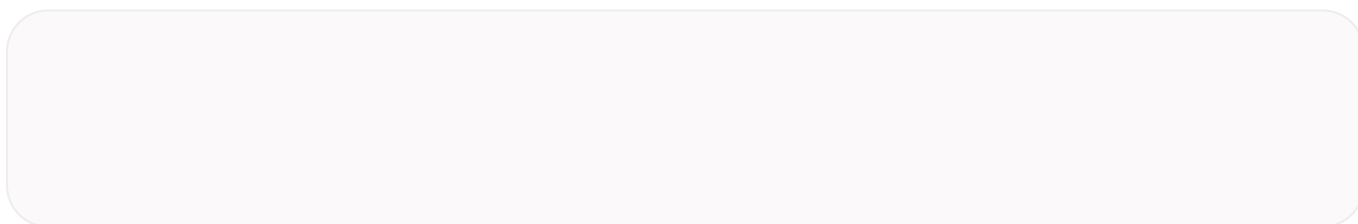
✓ Encabezado Expandible 2

Contenido del Encabezado Expandible 2

Encabezado 3

✓ Encabezado Expandible 3

Contenido del Encabezado Expandible 3



Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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",
}
```

Encabezado 1	Encabezado 2	Encabezado 3
Encabezado 1	Encabezado 2	Encabezado 3

Encabezado 1

Encabezado 2

Encabezado 3

✎ Encabezado Expandible 1

Contenido

✎ Encabezado Expandible 2

Contenido

✎ Encabezado Expandible 3

Contenido

Encabezado 1

Encabezado 2

Encabezado 3

▼ Encabezado Exp 1

Contenido

▼ Encabezado Exp 2

Contenido

▼ Encabezado Exp 3

Contenido

Encabezado 1

Encabezado 2

Encabezado 3

▼ Encabezado Exp 1
Contenido

▼ Encabezado Exp 2
Contenido

▼ Encabezado Exp 3
Contenido

Encabezado 1

2 Encabezado 2

Encabezado 2

3 Encabezado 3

Encabezado 3

4 Encabezado Expandible 1

Contenido

5 Encabezado Expandible 2

Contenido

6 Encabezado Expandible 3

Contenido

Obtener Pasteles

Obtener un pastel por su ID

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

Request

BODY PARAMETERS

id String **required**

ID del pastel a obtener

```
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": "Nombre del pastel",
}
```

✓ Encabezado expandible 1

Contenido

✓ Encabezado expandible 2

Contenido

✓ Encabezado expandible 3

Contenido

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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": "Nombre del pastel",
}
```

Paste an image link or upload your image

Image URL

Add Image

/

6.1. Fragmento de contenido de enlaces ancla

Encabezado 1

✓ Encabezado Expandible 1

Contenido del Encabezado Expandible 1

Encabezado 2

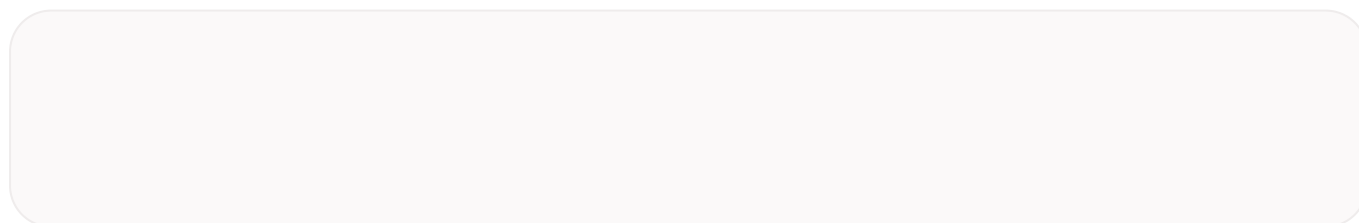
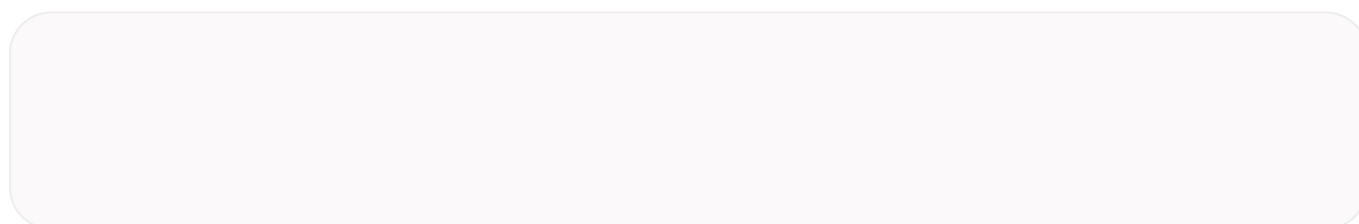
✓ Encabezado Expandible 2

Contenido del Encabezado Expandible 2

Encabezado 3

✓ Encabezado Expandible 3

Contenido del Encabezado Expandible 3



Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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": "Nombre del pastel",
}
```

Encabezado 1	Encabezado 2	Encabezado 3
Encabezado 1	Encabezado 2	Encabezado 3

Encabezado 1

Encabezado 2

Encabezado 3

✎ Encabezado Expandible 1

Contenido

✎ Encabezado Expandible 2

Contenido

✎ Encabezado Expandible 3

Contenido

Encabezado 1

Encabezado 2

Encabezado 3

✖ Encabezado Exp 1

Contenido

✖ Encabezado Exp 2

Contenido

✖ Encabezado Exp 3

Contenido

Encabezado 1

Encabezado 2

Encabezado 3

✖ Encabezado Exp 1
Contenido

✖ Encabezado Exp 2
Contenido

✖ Encabezado Exp 3
Contenido

Encabezado 1

2

Encabezado 2

Encabezado 2

3

Encabezado 3

Encabezado 3

4

✓ Encabezado Expandible 1

Contenido

5

✓ Encabezado Expandible 2

Contenido

6

✓ Encabezado Expandible 3

Contenido

Obtener Pasteles

Obtener un pastel por su ID

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

Request

BODY PARAMETERS

id String **required**

ID del pastel a obtener

```
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": "Nombre del pastel",
}
```

✓ Encabezado expandible 1

Contenido

✓ Encabezado expandible 2

Contenido

✓ Encabezado expandible 3

Contenido

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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",
}
```


7. Translating...

8. Todos los bloques

Encabezado 1 BOOBOO

✕ Encabezado Expandible 1

Contenido de Encabezado Expandible 1

Encabezado 2

✕ Encabezado Expandible 2

Contenido de Encabezado Expandible 2

Encabezado 3

✕ Encabezado Expandible 3

Contenido de Encabezado Expandible 3

- ☐ Lista de verificación
- ☐ Lista de verificación 2
- ☐ Lista de verificación 3

TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA

BOTÓN

Aviso

Haciendo

MINI TAREA 1

Probando

MINI TAREA 2

Completado

MINI TAREA 3



CUADRÍCULA DE ENLACES
1



CUADRÍCULA DE ENLACES
2



CUADRÍCULA DE ENLACES
3

- 1

Flujo de trabajo 1

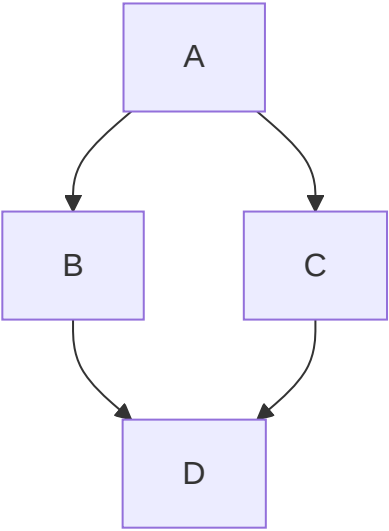
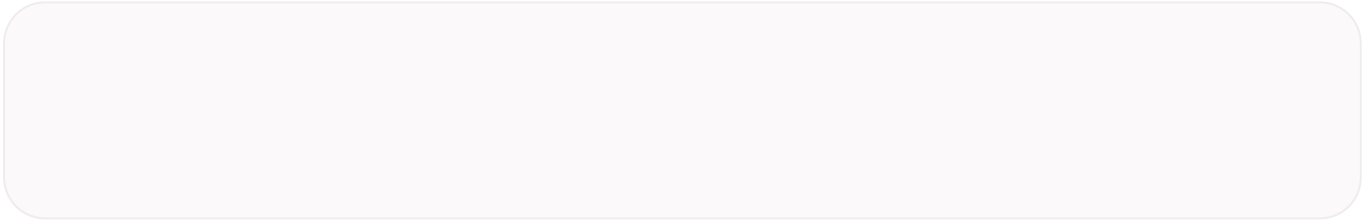
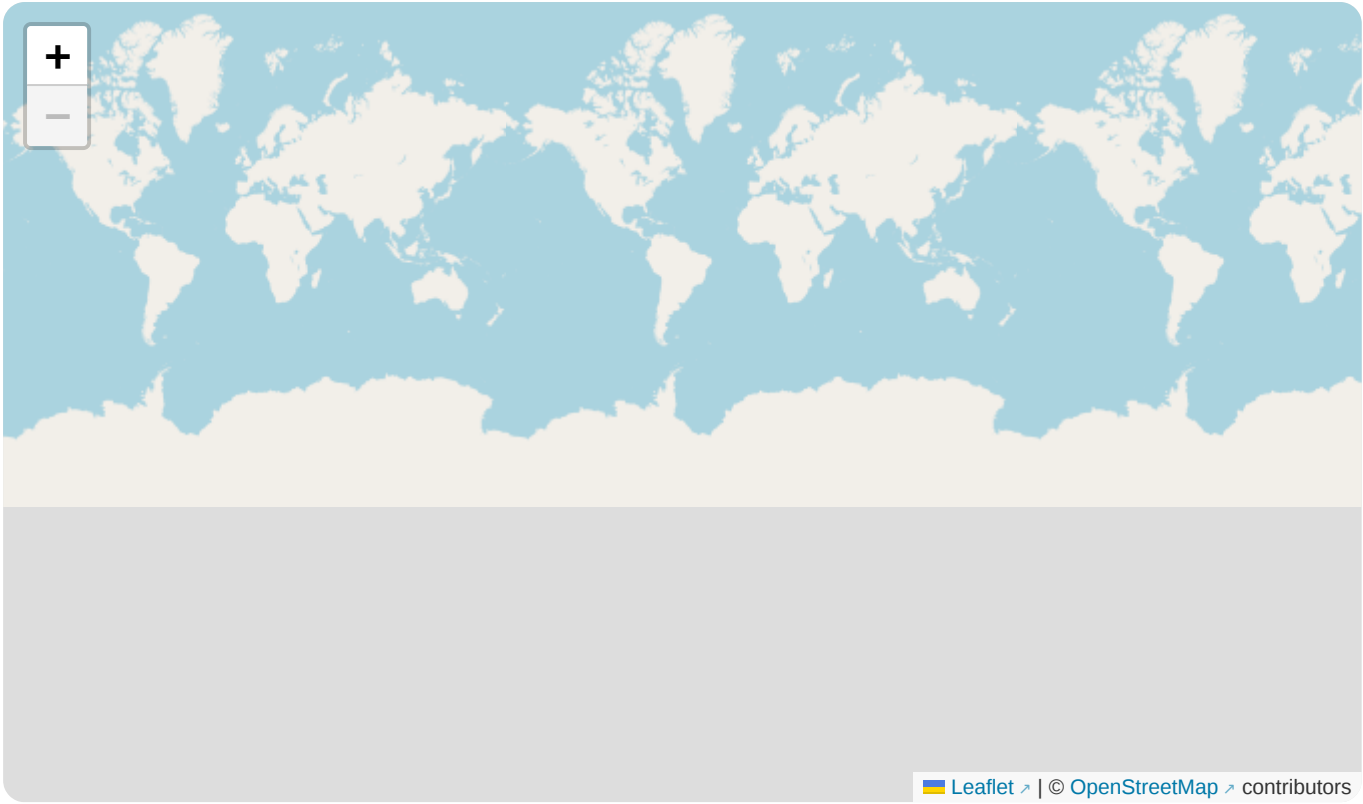
Contenido del flujo de trabajo 1
- 2

Flujo de trabajo 2

Contenido del flujo de trabajo 2

Contenido de la Pestaña 1

- Lista con viñeta 1
 - Lista con viñeta 2
 - Lista con viñeta 3
1. Lista numerada 1
 1. Lista numerada 2
 1. Lista numerada 3



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

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

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

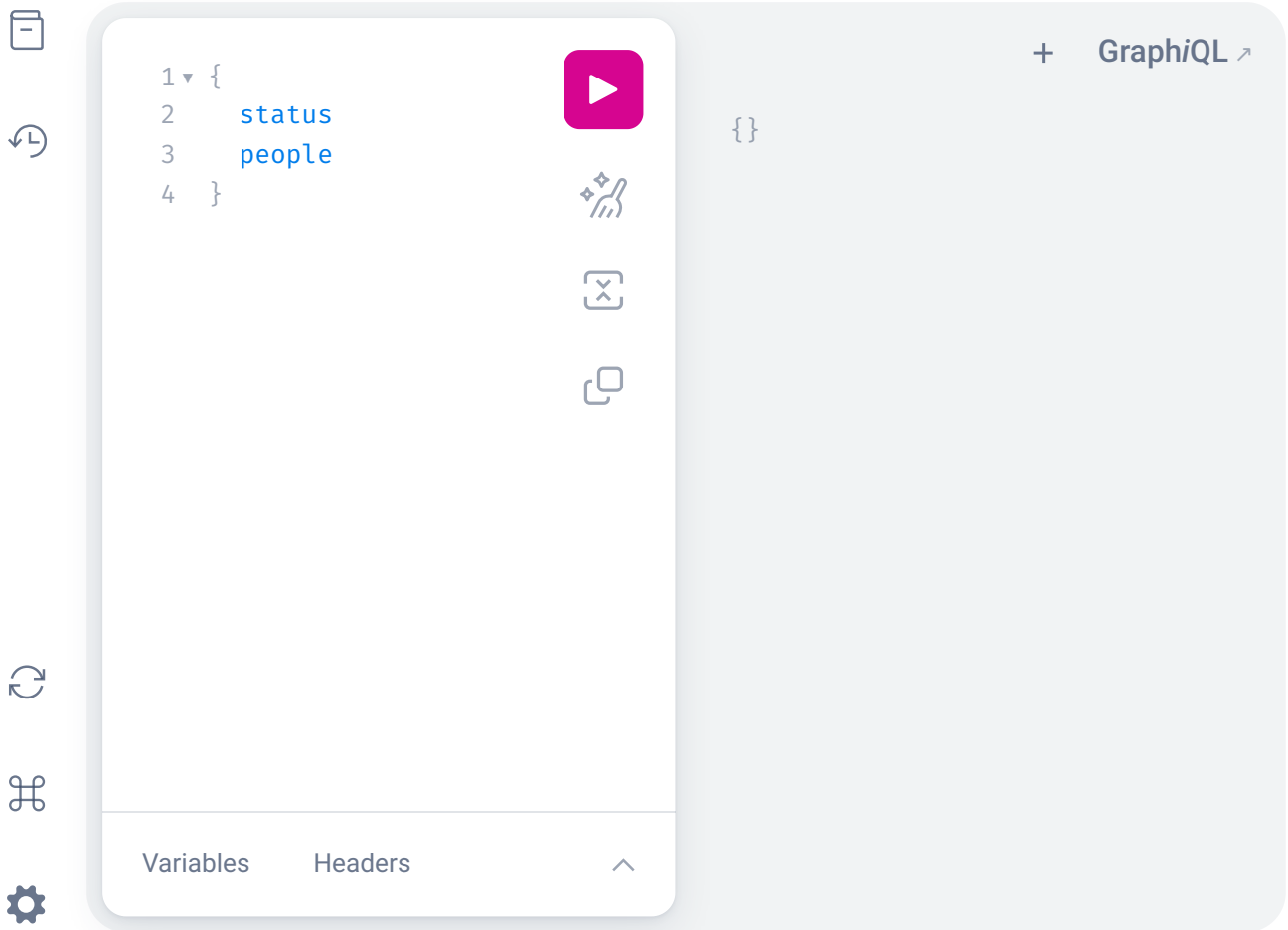
```
var raw = JSON.stringify({
  "id": "String"
});
```

```
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",
}
```

Loading swagger...



sadasdas

Added

Contenido 1

Fixed

Contenido 2

Added

Added

9. CATEGORÍA PARA REJILLAS DE ENLACES

9.1. DOCUMENTO DE CATEGORÍA

9.1.1. SOLO UN DOCUMENTO SIMPLE

9.2. PRIMER DOCUMENTO

ESTE ES EL CONTENIDO DEL PRIMER DOCUMENTO.

9.2.2. 1.2 DOCUMENT

9.2.4. DOCUMENTO OCULTO

9.3. DOCUMENTO OCULTO CON HIJO

9.3.1. Primer documento oculto

9.4. Segundo documento

ESTE ES EL CONTENIDO DEL SEGUNDO DOCUMENTO.

9.4.1. 2.1 DOCUMENTO

9.4.4. DOCUMENTO OCULTO

9.5. TERCER DOCUMENTO

ESTE ES EL CONTENIDO DEL TERCER DOCUMENTO.

9.5.1. 3.1 DOCUMENTO

9.5.4. Documento oculto

9.6. Sin título

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 (clonado)

Encabezado 1

▼ Encabezado Expandible 1

Contenido del Encabezado Expandible 1

Encabezado 2

▼ Encabezado Expandible 2

Contenido del Encabezado Expandible 2

Encabezado 3

▼ Encabezado Expandible 3

Contenido del Encabezado Expandible 3

- ☐ Lista de verificación
- ☐ Lista de verificación 2
- ☐ Lista de verificación 3

TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA

BOTÓN

Aviso

Haciendo

MINI TAREA 1

Probando

MINI TAREA 2

Completo

MINI TAREA 3

CUADRÍCULA DE ENLACES
1

CUADRÍCULA DE ENLACES
2

CUADRÍCULA DE ENLACES
3

- 1

Flujo de trabajo 1

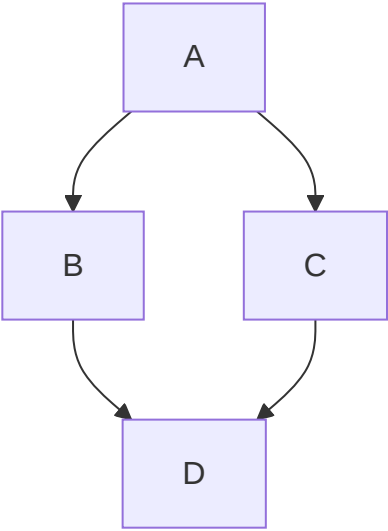
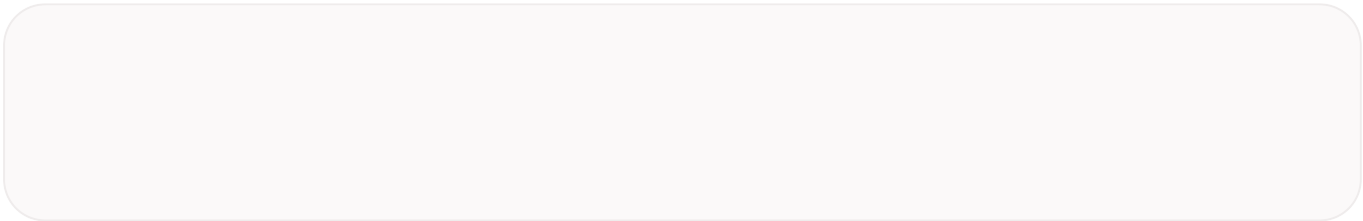
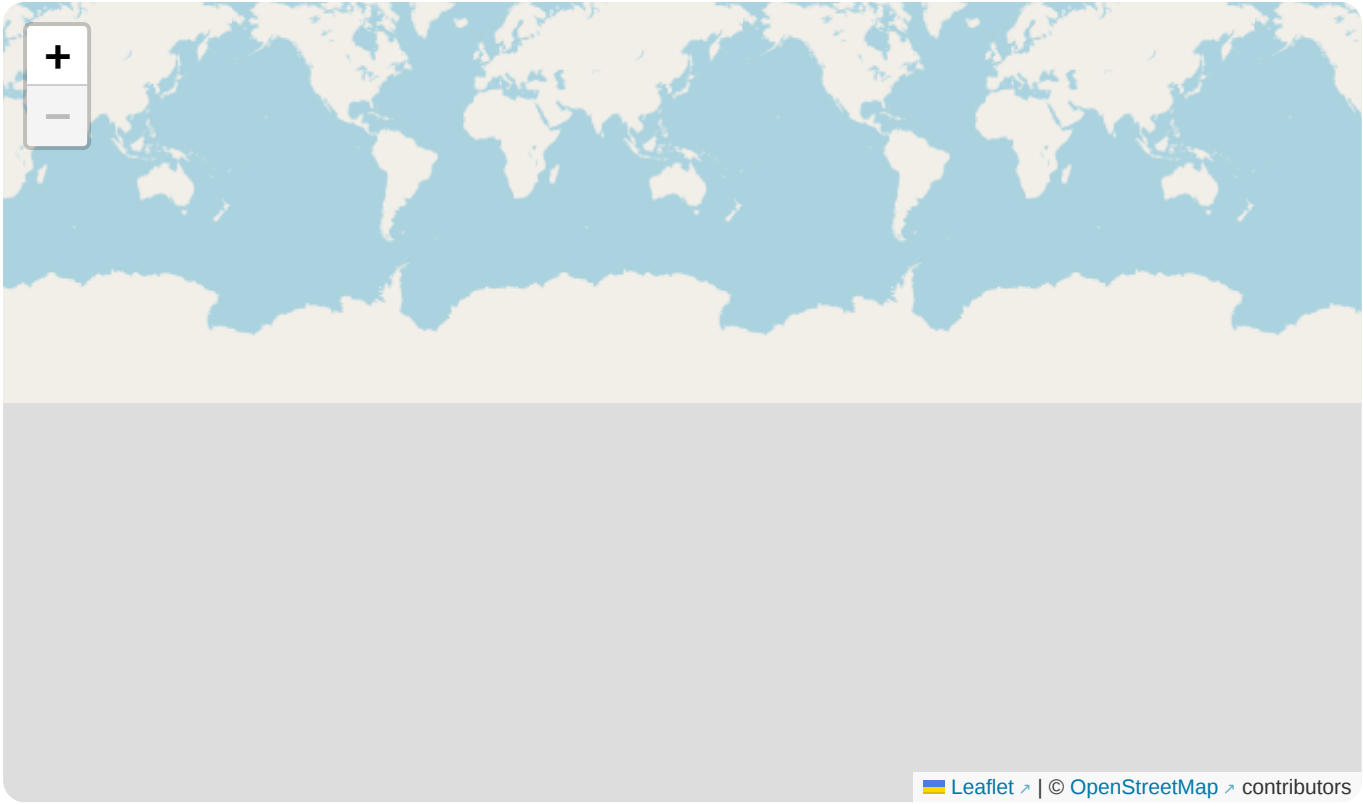
Contenido del flujo de trabajo 1
- 2

Flujo de trabajo 2

Contenido del flujo de trabajo 2

Contenido de la Pestaña 1

- Lista con viñetas 1
 - Lista con viñetas 2
 - Lista con viñetas 3
1. Lista numerada 1
 1. Lista numerada 2
 1. Lista numerada 3



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

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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 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
object
(body)

order placed for purchasing the pet

Example Value | Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:57:00.138Z",
  "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": "2026-06-05T08:57:00.245Z", "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": "2026-06-05T08:57:00.249Z",  
"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
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 ↗

{}

Título del registro de cambios

Added

Contenido 1

Fixed

Contenido 2

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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. CODE DRAWERS

Encabezado 1

▼ Encabezado Expandible 1

Contenido del Encabezado Expandible 1

Encabezado 2

▼ Encabezado Expandible 2

Contenido del Encabezado Expandible 2

Encabezado 3

▼ Encabezado Expandible 3

Contenido del Encabezado Expandible 3

- ☐ Lista de verificación
- ☐ Lista de verificación 2
- ☐ Lista de verificación 3

TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA

BOTÓN

Aviso

Doing

MINI TASK 1

Testing

MINI TASK 2

Done

MINI TASK 3

CUADRÍCULA DE ENLACES
1

CUADRÍCULA DE ENLACES
2

CUADRÍCULA DE ENLACES
3

- 1

Flujo de trabajo 1

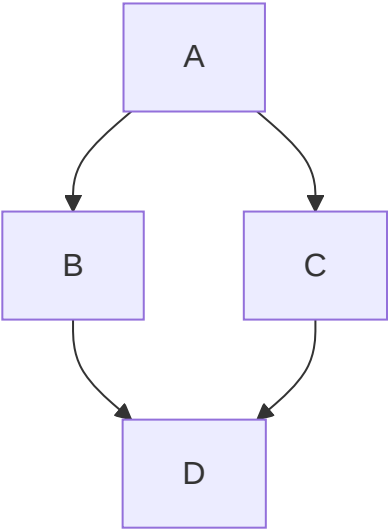
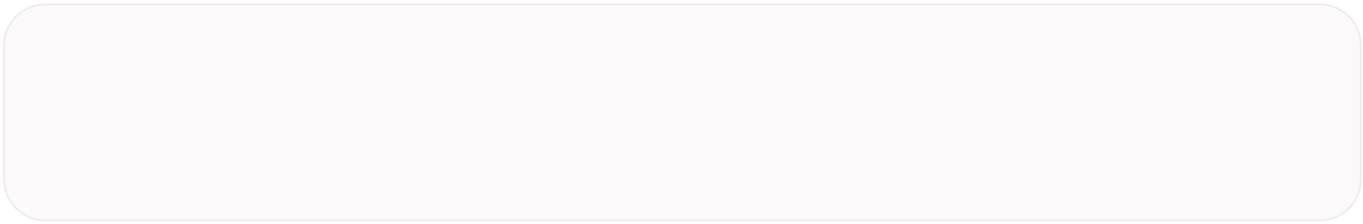
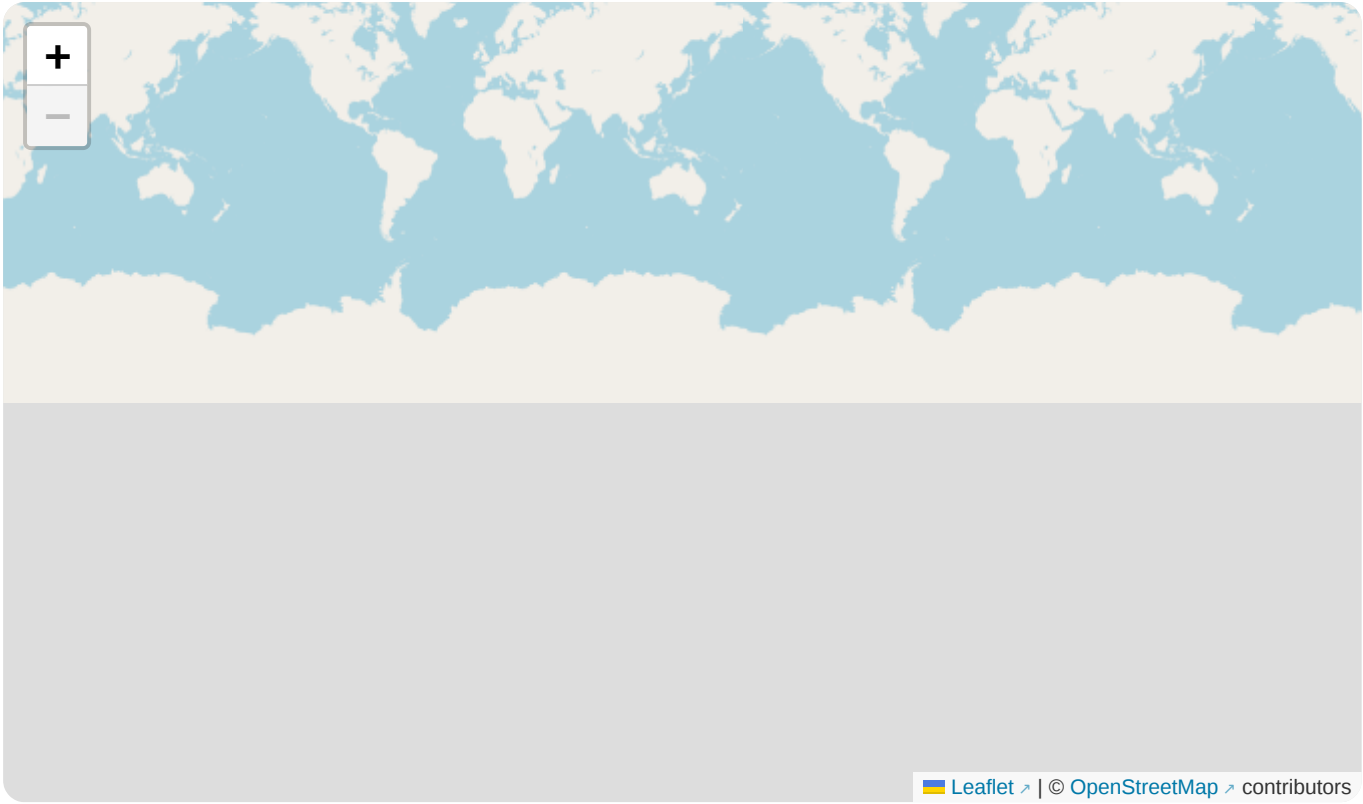
Contenido del flujo de trabajo 1
- 2

Flujo de trabajo 2

Contenido del flujo de trabajo 2

Contenido de la Pestaña 1

- Lista con viñetas 1
 - Lista con viñetas 2
 - Lista con viñetas 3
1. Lista numerada 1
 1. Lista numerada 2
 1. Lista numerada 3



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

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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> <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 updated**integer(\$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
object
(body)

order placed for purchasing the pet

Example Value | Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:57:00.146Z",
  "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": "2026-06-05T08:57:00.353Z", "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": "2026-06-05T08:57:00.359Z",  
"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

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

GET

/user/{username} Get user by user name



Parameters

Try it out

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

username * required

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

string
(path)

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 ↗

{}

Título del registro de cambios

Added

Contenido 1

Fixed

Contenido 2

Get Cakes

Obtener un pastel por su ID

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID del pastel a obtener

```
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. BLOQUES INDIVIDUALES

12.1. Encabezado 1

Encabezado 1

12.2. Encabezado 2

Encabezado 2

12.3. Encabezado 3

Encabezado 3

13. ENLACE CUADRÍCULAS TARJETAS

PRIMER DOCUMENTO

SEGUNDO DOCUMENTO

TERCER DOCUMENTO

14. ENLACE CUADRÍCULAS NORMAL

PRIMER DOCUMENTO

SEGUNDO DOCUMENTO

TERCER DOCUMENTO

15. Preguntas y respuestas

Solo un documento.

Haga clic para editar la etiqueta del botón

Haga clic para editar la etiqueta del botón

16. ALGUNOS BLOQUES

Encabezado 1

▼ Encabezado expandible 1

Contenido del encabezado expandible 1

Encabezado 2

▼ Encabezado expandible 2

Contenido del encabezado expandible 2

Encabezado 3

▼ Encabezado expandible 3

Contenido del encabezado expandible 3

- ☐ Lista de verificación
- ☐ Lista de verificación 2
- ☐ Lista de verificación 3

TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA
TABLA	TABLA	TABLA

BOTÓN

Aviso

Doing

MINI TASK 1

Testing

MINI TASK 2

Done

MINI TASK 3



CUADRÍCULA DE ENLACES
1



CUADRÍCULA DE ENLACES
2



CUADRÍCULA DE ENLACES
3

- 1

Flujo de trabajo 1

Contenido del flujo de trabajo 1
- 2

Flujo de trabajo 2

Contenido del flujo de trabajo 2

PESTAÑA 1

PESTAÑA 2

Contenido de la pestaña 1

• Lista con viñetas 1

◦ Lista con viñetas 2

▪ Lista con viñetas 3

1. Lista numerada 1

1. Lista numerada 2

1. Lista numerada 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. API de muestra

18.1. Listar todos los ítems.

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. Agregar un elemento.

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
```