

GERMAN ↗

GERMAN

1. [CONFLUENCE] DOKUMENT FÜR IMPORTE
2. link
3. Translating...
4. SWAGGER
5. API-Endpunkte
6. Anker-Links 
 - 6.1. Ankerlinks-Inhaltsausschnitt
7. Translating...
8. Alle Blöcke
9. KATEGORIE FÜR LINK-GITTER
 - 9.1. KATEGORIE DOKUMENT
 - 9.1.1. NUR EIN EINFACHES DOKUMENT
 - 9.2. ERSTES DOKUMENT
 - 9.2.1. 1.1 DOKUMENT
 - 9.2.2. 1.2 DOKUMENT
 - 9.2.3. 1.3 DOCUMENT
 - 9.2.4. VERSTECKTES DOKUMENT
 - 9.3. VERSTECKTES DOKUMENT MIT KIND
 - 9.3.1. ERSTES VERBORGENES DOKUMENT
 - 9.4. ZWEITES DOKUMENT
 - 9.4.1. 2.1 DOCUMENT
 - 9.4.2. 2.2 DOCUMENT
 - 9.4.3. 2.3 DOCUMENT
 - 9.4.4. VERSTECKTES DOKUMENT
 - 9.5. Drittes Dokument
 - 9.5.1. 3.1 DOCUMENT
 - 9.5.2. 3.2 DOKUMENT
 - 9.5.3. 3.3 DOKUMENT

9.5.4. VERSTECKTES DOKUMENT

9.6. Ohne Titel

10. CODE DRAWERS (cloned)
11. CODE DRAWERS
12. Individuelle Blöcke
 - 12.1. Überschrift 1
 - 12.2. Überschrift 2
 - 12.3. Überschrift 3
13. LINK-GITTER-KARTEN
14. LINK GRIDS NORMAL
15. FRAGEN UND ANTWORTEN
16. Einige Blöcke
17. TEX
18. Beispiel-API
 - 18.1. Alle Artikel auflisten.
 - 18.2. Einen Eintrag hinzufügen.

1. [CONFLUENCE] DOKUMENT FÜR IMPORTE

Hey dies ist ein Dokument, das den Benutzer korrekt pingen sollte.

Hey dies ist ein Dokument, das das Dokument korrekt pingen sollte [CONFLUENCE] DOC FOR MENTION ↗ korrekt.

Hey dies ist ein Dokument, das einen Hyperlink zu dieser Seite ↗.

Dies ist ein LINK ↗

Dies ist ein weiterer LINK ↗

2. link

Anker-Links ☐☐

3. Translating...

4. SWAGGER

bla bla

Get Cakes eeee

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

Response

BODY PARAMETERS

id	String	required
----	--------	----------

ID des abzurufenden Kuchens

Parameter name	Integer	required
----------------	---------	----------

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

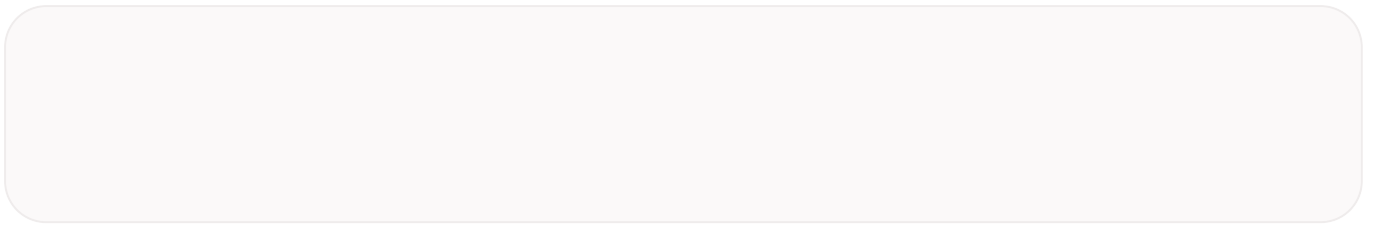
var raw = JSON.stringify({
  "id": "string"
});

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. API-Endpunkte



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. Anker-Links ☐☐

Überschrift 1

✓ Ausklappbare Überschrift 1

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

✓ Ausklappbare Überschrift 2

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

✓ Ausklappbare Überschrift 3

Inhalt der ausklappbaren Überschrift 3

Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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": "Name des Kuchens",
}
```

Überschrift 1	Überschrift 2	Überschrift 3
Überschrift 1	Überschrift 2	Überschrift 3

Überschrift 1

Überschrift 2

Überschrift 3

✖ Ausklappbare Überschrift 1

Inhalt

✖ Ausklappbare Überschrift 2

Inhalt

✖ Ausklappbare Überschrift 3

Inhalt

Überschrift 1

Überschrift 2

Überschrift 3

✖ Ausklappbare
Überschrift 1

Inhalt

✖ Ausklappbare Überschrift
2

Inhalt

✖ Ausklappbare Überschrift 3

Inhalt

Überschrift 1

Überschrift 2

Überschrift 3

✓ Ausklapp- Überschrift 1

Inhalt

✓ Ausklapp- Überschrift 2

Inhalt

✓ Ausklapp- Überschrift 3

Inhalt

1 Überschrift 1

Überschrift 1

2 Überschrift 2

Überschrift 2

3 Überschrift 3

Überschrift 3

4 ✓ Ausklapp-Überschrift 1

Inhalt

5

✓ Ausklapp-Überschrift 2

Inhalt

6

✓ Ausklapp-Überschrift 3

Inhalt

Get Cakes

Einen Cake anhand seiner ID abrufen

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

Request

BODY PARAMETERS

id String **required**

ID des abzurufenden Cake

```
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",
}
```

✓ Ausklappbare Überschrift 1

Inhalt

✓ Ausklappbare Überschrift 2

Inhalt

✓ Ausklappbare Überschrift 3

Inhalt

Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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": "Name des Kuchens",
}
```

Paste an image link or upload your image

Image URL

Add Image

/

6.1. Ankerlinks-Inhaltsausschnitt

Überschrift 1

✓ Ausklappbare Überschrift 1

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

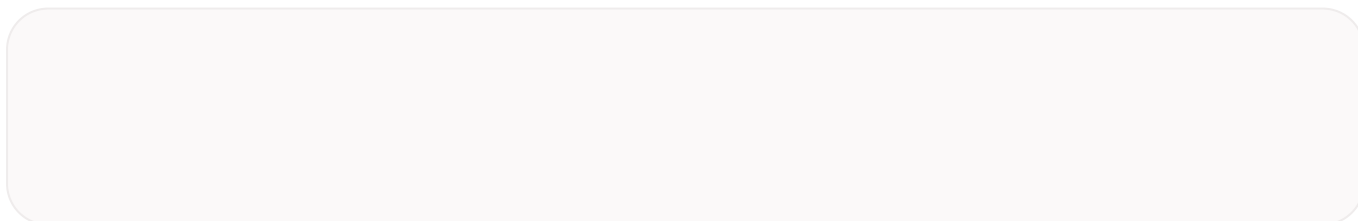
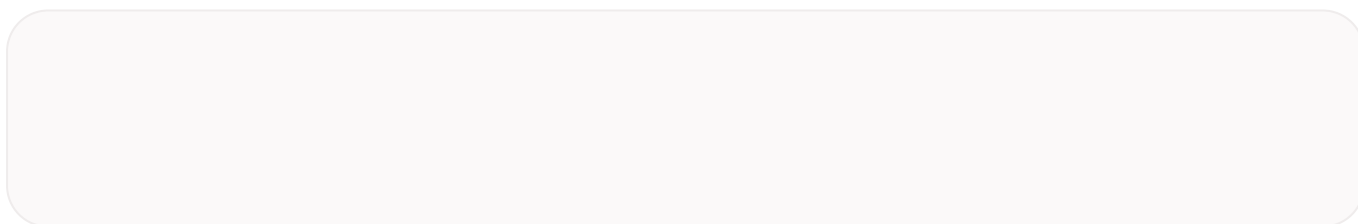
✓ Ausklappbare Überschrift 2

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

✓ Ausklappbare Überschrift 3

Inhalt der ausklappbaren Überschrift 3



Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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",
}
```

Überschrift 1	Überschrift 2	Überschrift 3
Überschrift 1	Überschrift 2	Überschrift 3

Überschrift 1

Überschrift 2

Überschrift 3

✖ Ausklappbare Überschrift 1

Inhalt

✖ Ausklappbare Überschrift 2

Inhalt

✖ Ausklappbare Überschrift 3

Inhalt

Überschrift 1

Überschrift 2

Überschrift 3

✓ **Erweiterte Überschrift
1**

Inhalt

✓ **Erweiterte Überschrift 2**

Inhalt

✓ **Erweiterte Überschrift 3**

Inhalt

Überschrift 1

Überschrift 2

Überschrift 3

▼ **Erweiterte Überschrift 1**

Inhalt

▼ **Erweiterte Überschrift 2**

Inhalt

▼ **Erweiterte Überschrift 3**

Inhalt

1

Überschrift 1

Überschrift 1

2

Überschrift 2

Überschrift 2

3

Überschrift 3

Überschrift 3

4

▼ **Erweiterte Überschrift 1**

Inhalt

5

✓ **Erweiterte Überschrift 2**

Inhalt

6

✓ **Erweiterte Überschrift 3**

Inhalt

Get Cakes

Einen Kuchen anhand seiner ID abrufen

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

Request

BODY PARAMETERS

id String **required**

ID des abzurufenden Kuchens

```
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",
}
```

✓ Erweiterbare Überschrift 1

Inhalt

✓ Erweiterbare Überschrift 2

Inhalt

✓ Erweiterbare Überschrift 3

Inhalt

Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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. Alle Blöcke

Überschrift 1 BOOBOO

✕ Ausklappbare Überschrift 1

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

✕ Ausklappbare Überschrift 2

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

✕ Ausklappbare Überschrift 3

Inhalt der ausklappbaren Überschrift 3

- ☐ Checkliste
- ☐ Checkliste 2
- ☐ Checkliste 3

TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE

SCHALTFLÄCHE

Hinweis

In Arbeit

MINI AUFGABE 1

Testen

MINI AUFGABE 2

Erledigt

MINI AUFGABE 3



LINK GITTER 1



LINK GITTER 2



LINK GITTER 3

- 1

Workflow 1

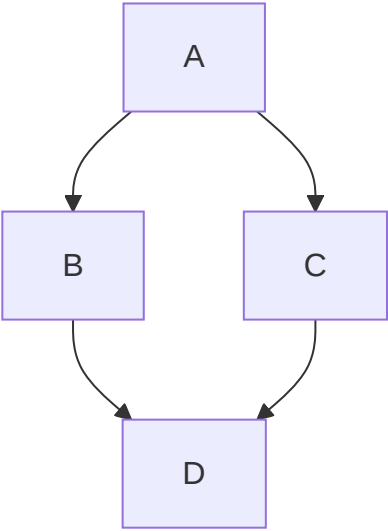
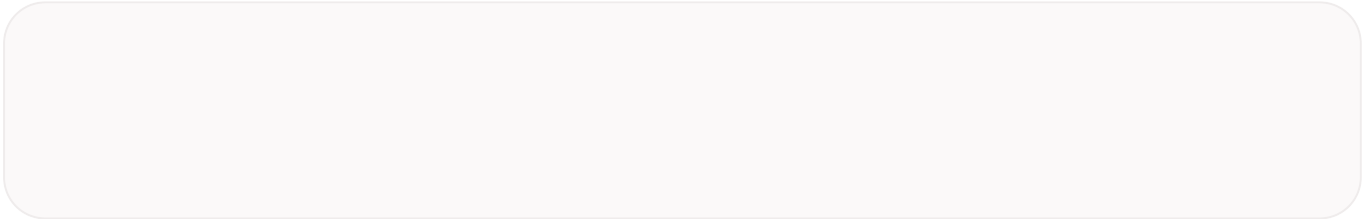
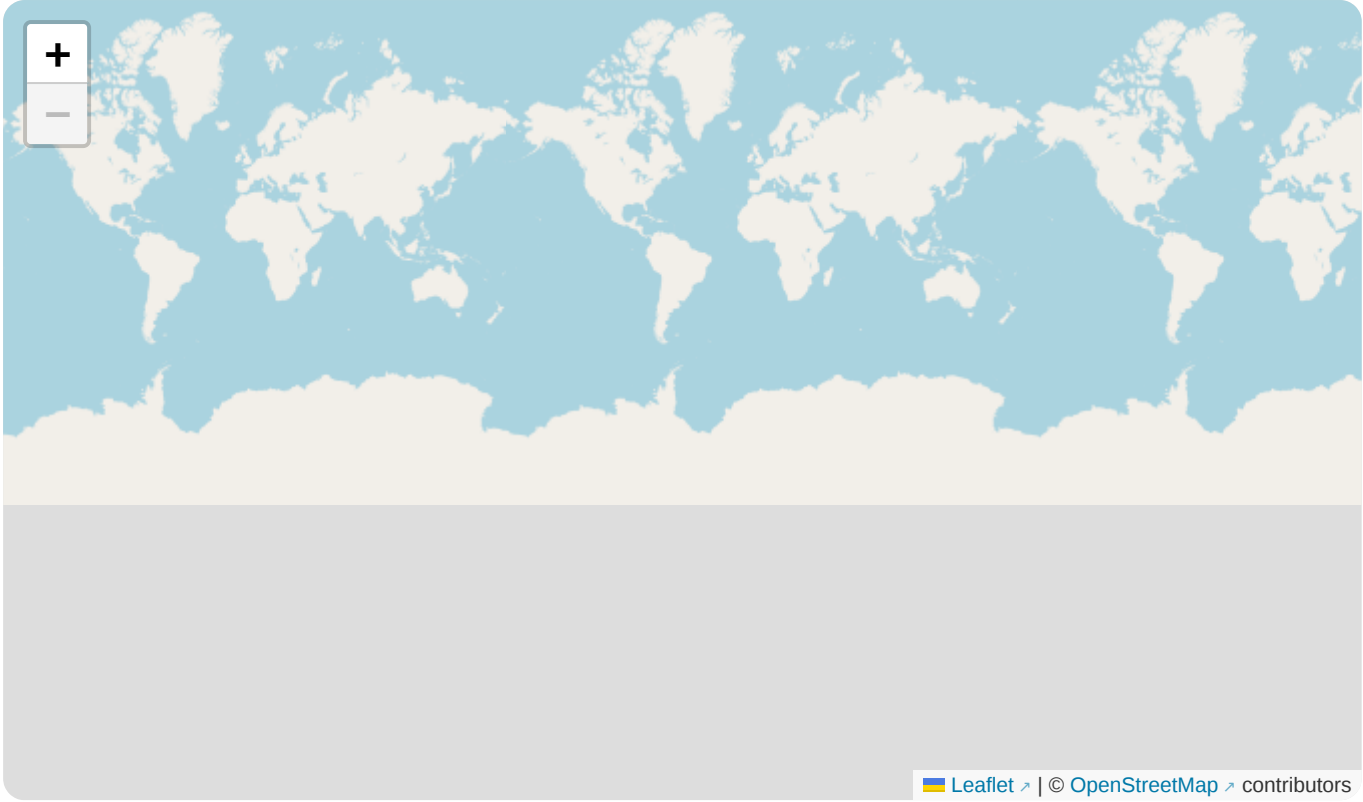
Inhalt von Workflow 1
- 2

Workflow 2

Inhalt von Workflow 2

Inhalt von Reiter 1

- Aufzählungspunkt 1
 - Aufzählungspunkt 2
 - Aufzählungspunkt 3
- Nummerierte Liste 1
 - Nummerierte Liste 2
 - Nummerierte Liste 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 <code>array<string></code> (query)	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 Pet id to delete

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": "2026-06-05T08:41:33.395Z",
  "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": "2026-06-05T08:41:33.474Z", "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": "2026-06-05T08:41:33.481Z", "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

```
[
  {
    "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 ↗

{}

sadasdas

Added

Content 1

Fixed

Content 2

Added

Added

9. KATEGORIE FÜR LINK- GITTER

9.1. KATEGORIE DOKUMENT

9.1.1. NUR EIN EINFACHES DOKUMENT

9.2. ERSTES DOKUMENT

DIES IST DER INHALT DES ERSTEN DOKUMENTS.

9.2.1. 1.1 DOKUMENT

9.2.2. 1.2 DOKUMENT

9.2.3. 1.3 DOCUMENT

9.2.4. VERSTECKTES DOKUMENT

9.3. VERSTECKTES DOKUMENT MIT KIND

9.3.1. ERSTES VERBORGENES DOKUMENT

9.4. ZWEITES DOKUMENT

DIES IST DER INHALT DES ZWEITEN DOKUMENTS.

9.4.2. 2.2 DOCUMENT

9.4.4. VERSTECKTES DOKUMENT

9.5. Drittes Dokument

DIES IST DER INHALT DES DRITTEN DOKUMENTS.

9.5.4. VERSTECKTES DOKUMENT

9.6. Ohne Titel

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 (cloned)

Überschrift 1

▼ **Ausklappbare Überschrift 1**

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

▼ **Ausklappbare Überschrift 2**

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

▼ **Ausklappbare Überschrift 3**

Inhalt der ausklappbaren Überschrift 3

- ☐ Checkliste
- ☐ Checkliste 2
- ☐ Checkliste 3

TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE

TASTE

Hinweis

Doing

MINI-AUFGABE 1

Testing

MINI-AUFGABE 2

Done

MINI-AUFGABE 3

LINK-GITTER 1

LINK-GITTER 2

LINK-GITTER 3

- 1

Workflow 1

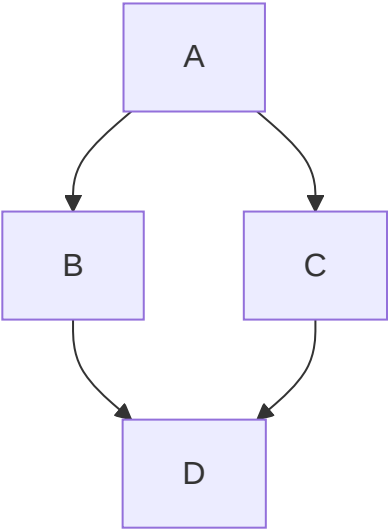
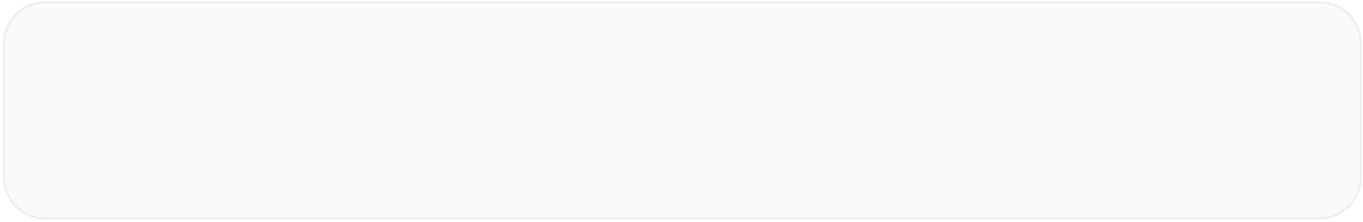
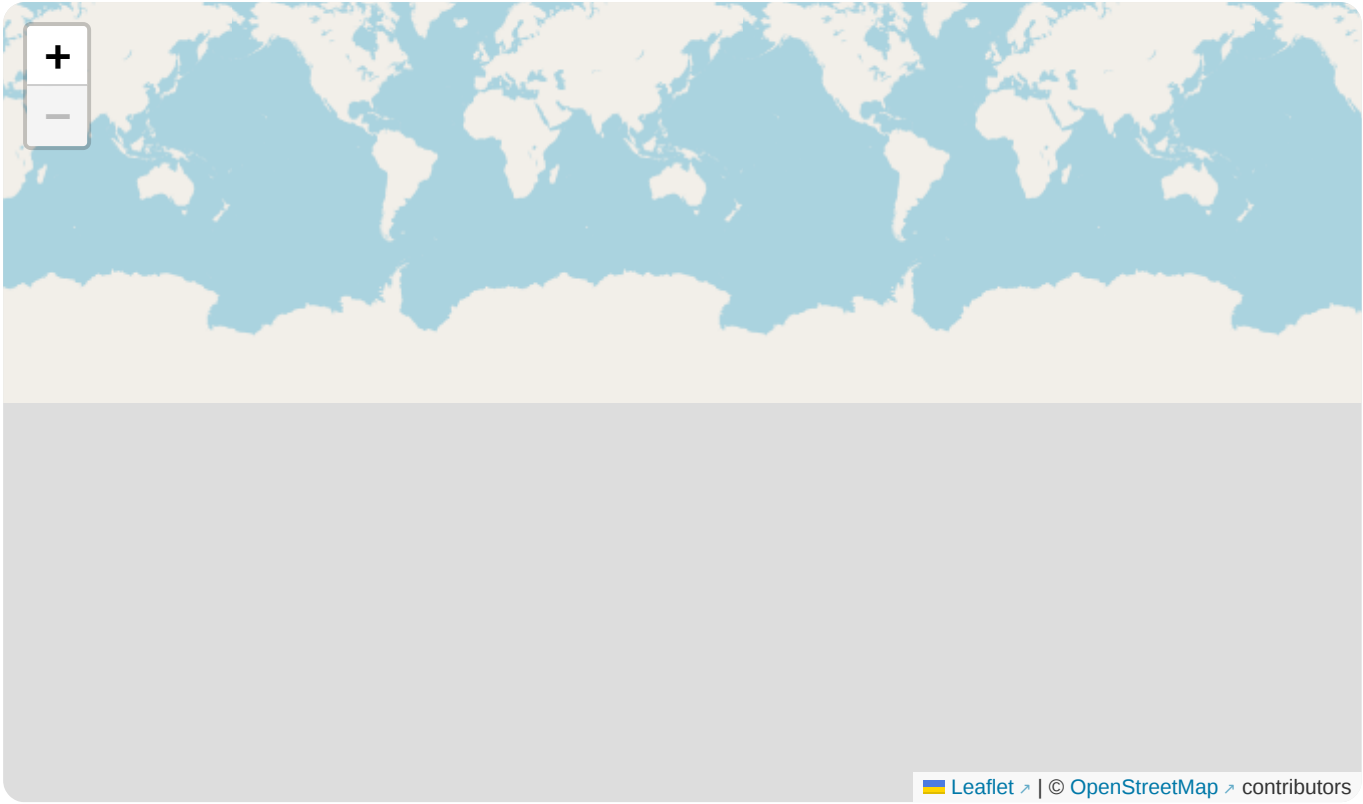
Inhalt von Workflow 1
- 2

Workflow 2

Inhalt von Workflow 2

Inhalt von Reiter 1

- Aufzählungspunkt 1
 - Aufzählungspunkt 2
 - Aufzählungspunkt 3
- Nummerierte Liste 1
 - Nummerierte Liste 2
 - Nummerierte Liste 3



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

Kuchen abrufen

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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>
(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 (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": "2026-06-05T08:43:02.671Z",
  "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": "2026-06-05T08:43:02.779Z", "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": "2026-06-05T08:43:02.784Z", "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

```
[
  {
    "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
(path)

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 >

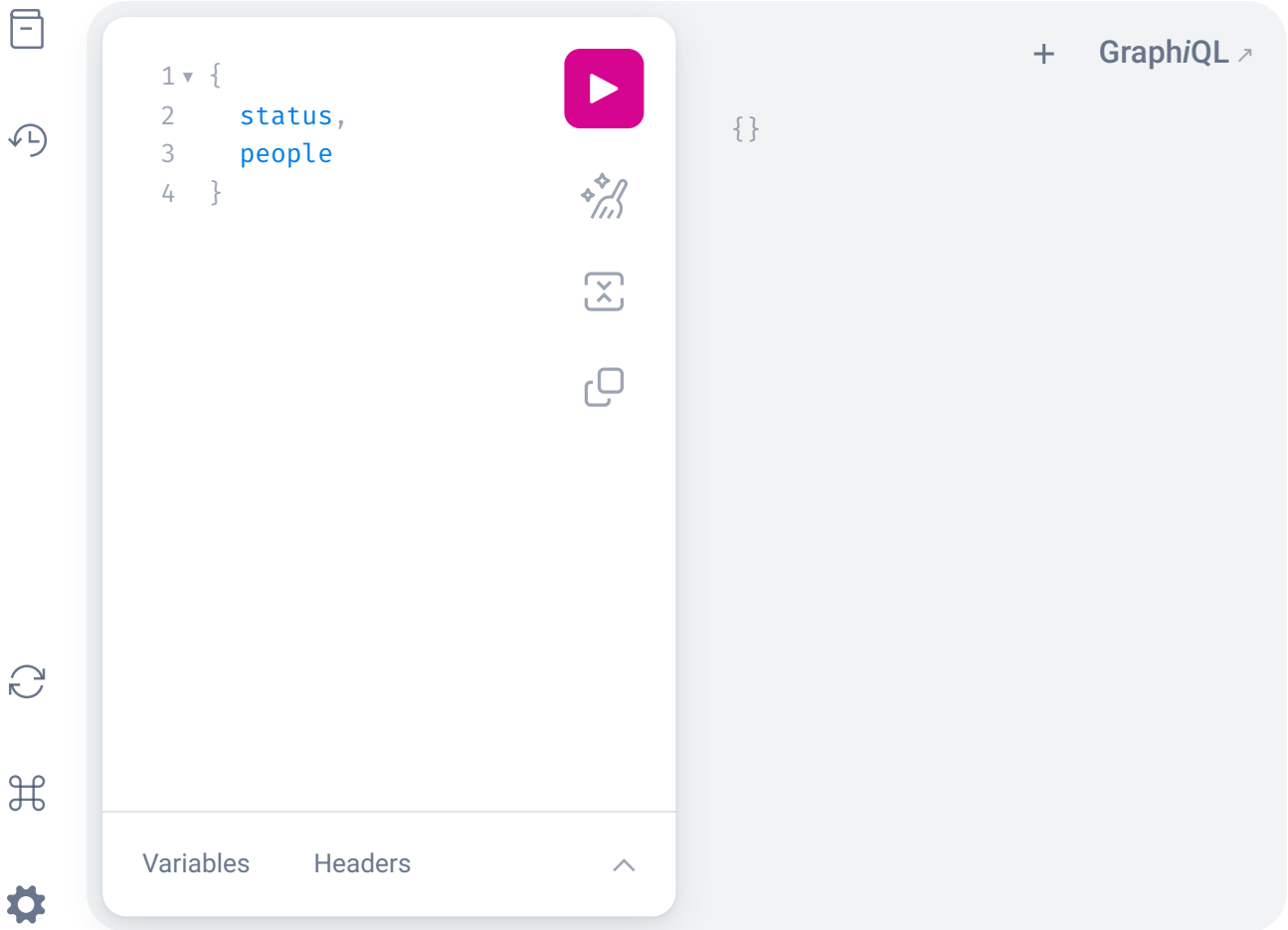
Category >

Pet >

Tag >

Order >

User >



Changelog-Titel

Added

Inhalt 1

Fixed

Inhalt 2

Kuchen abrufen

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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",
}
```


Kuchen abrufen

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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

Überschrift 1

▼ **Ausklappbare Überschrift 1**

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

▼ **Ausklappbare Überschrift 2**

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

▼ **Ausklappbare Überschrift 3**

Inhalt der ausklappbaren Überschrift 3

- ☐ Checkliste
- ☐ Checkliste 2
- ☐ Checkliste 3

TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE

SCHALTFLÄCHE

Hinweis

In Bearbeitung

MINI AUFGABE 1

Testen

MINI AUFGABE 2

Erledigt

MINI AUFGABE 3



LINK GITTER 1



LINK GITTER 2



LINK GITTER 3

- 1

Workflow 1

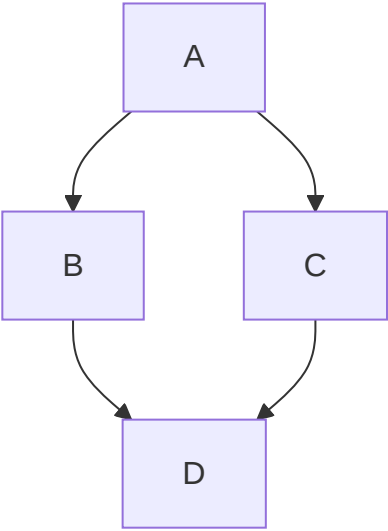
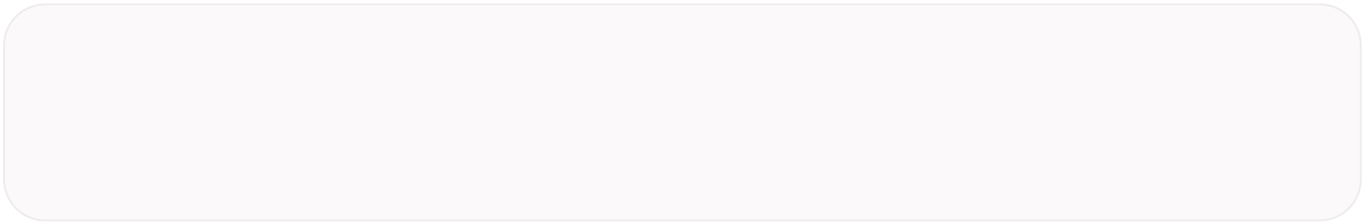
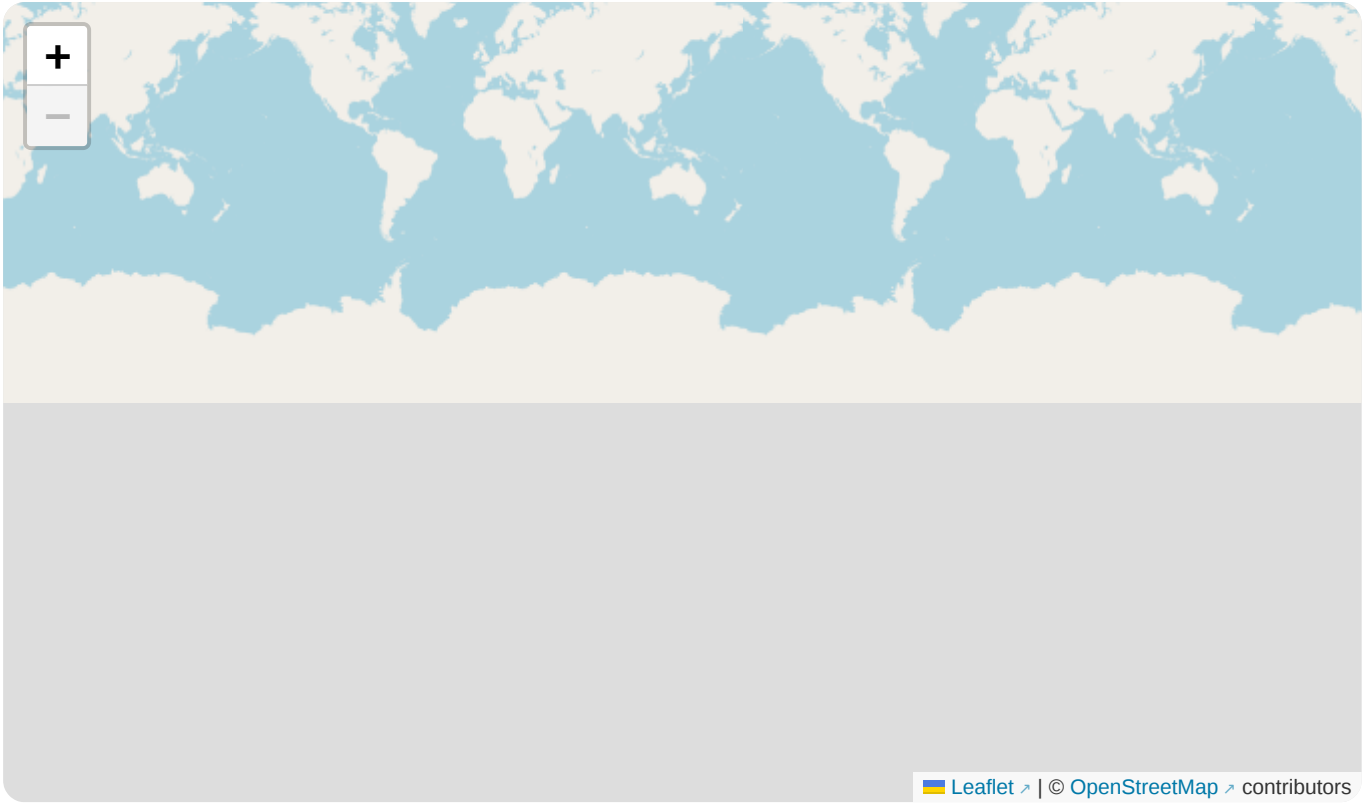
Inhalt von Workflow 1
- 2

Workflow 2

Inhalt von Workflow 2

Inhalt von Reiter 1

- Aufzählungspunkt 1
 - Aufzählungspunkt 2
 - Aufzählungspunkt 3
- Nummerierte Liste 1
 - Nummerierte Liste 2
 - Nummerierte Liste 3



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

Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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> (query)	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)

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
object
(body)

order placed for purchasing the pet

Example Value | Model

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2026-06-05T08:43:03.952Z",
  "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": "2026-06-05T08:43:03.998Z", "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": "2026-06-05T08:43:04.001Z", "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

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

```
"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

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 ↗

{}

Changelog-Titel

Added

Inhalt 1

Fixed

Inhalt 2

Get Cakes

Einen Kuchen anhand seiner ID abrufen

GET

https://api.cakes.com



Request

BODY PARAMETERS

id String required

ID des abzurufenden Kuchens

```
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. Individuelle Blöcke

12.1. Überschrift 1

Überschrift 1

12.2. Überschrift 2

Überschrift 2

12.3. Überschrift 3

Überschrift 3

13. LINK-GITTER-KARTEN

ERSTES DOKUMENT

ZWEITES DOKUMENT

DRITTES DOKUMENT

14. LINK GRIDS NORMAL

ERSTES DOKUMENT

ZWEITES DOKUMENT

DRITTES DOKUMENT

15. FRAGEN UND ANTWORTEN

Nur ein Dokument.

Klicken Sie, um die Beschriftung der Schaltfläche zu bearbeiten

Klicken Sie, um die Beschriftung der Schaltfläche zu bearbeiten

16. Einige Blöcke

Überschrift 1

▼ **Ausklappbare Überschrift 1**

Inhalt der ausklappbaren Überschrift 1

Überschrift 2

▼ **Ausklappbare Überschrift 2**

Inhalt der ausklappbaren Überschrift 2

Überschrift 3

▼ **Ausklappbare Überschrift 3**

Inhalt der ausklappbaren Überschrift 3

- ☐ Checkliste
- ☐ Checkliste 2
- ☐ Checkliste 3

TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE
TABELLE	TABELLE	TABELLE

SCHALTFLÄCHE

Hinweis

VERTIKALE TEILUNG, LINKE SEITE

VERTIKALE TEILUNG, RECHTE SEITE

In Arbeit

MINIAUFGABE 1



LINK GITTER 1

Testen

MINIAUFGABE 2



LINK GITTER 2

Erledigt

MINIAUFGABE 3



LINK GITTER 3

- 1

Workflow 1

Inhalt von Workflow 1
- 2

Workflow 2

Inhalt von Workflow 2

REGISTERKARTE 1

REGISTERKARTE 2

Inhalt von Registerkarte 1

- Aufzählungspunkt 1
 - Aufzählungspunkt 2
 - Aufzählungspunkt 3
1. Nummerierte Liste 1

1. Nummerierte Liste 2

1. Nummerierte Liste 3

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

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

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

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

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

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

$$\eta(T) = \begin{cases} 1, & \text{if } 30^\circ\text{C} \leq T \leq 35^\circ\text{C} \\ 0.8, & \text{if } 25^\circ\text{C} \leq T < 30^\circ\text{C} \\ 0, & \text{if } T < 25^\circ\text{C} \end{cases}$$

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

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

$$m_{\text{yeast}} = f_1(\mathbf{x}), \quad Y_{\text{fermenter}} = f_2(\mathbf{x})$$

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

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

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

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

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. Beispiel-API

18.1. Alle Artikel auflisten.

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. Einen Eintrag hinzufügen.

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
```