API MANUAL

this is a manual to 

What is an API ?

API stands for Application Programming Interface.

It consists of a web interface between the user and the database: it exposes url (called **endpoint**) to access, update, delete, create the data inside the tables of the database called resources providing HTTP **methods** using **parameters** in the url.

The API interface gives a facility for researcher and administrator to: access, consult and edit the data stored in the database. API is then the main entrance to control the settings, results and statistics of the database and understand the data underlying feeding each plot and graphic in the web dashboard.

Endpoint consist of a dynamic url for a resource accessible throught ***methods*** corresponding to actions, and ***parameters*** acting like filters.

For better understanding, they are organized into ***categories***. One categorie gathers multiple endpoints for one type of resource.

> Current API interface is available at :
>
> https://research.ludoeducation.fr

Anatomy of the API

The API provides an web user-friendly interface using Swagger and Open API specifications for describing the schema. In this interface, ***categories*** consists of dropdown list with a short description of the kind of resource. When you click on it: the full list of ***endpoints*** available will appear.

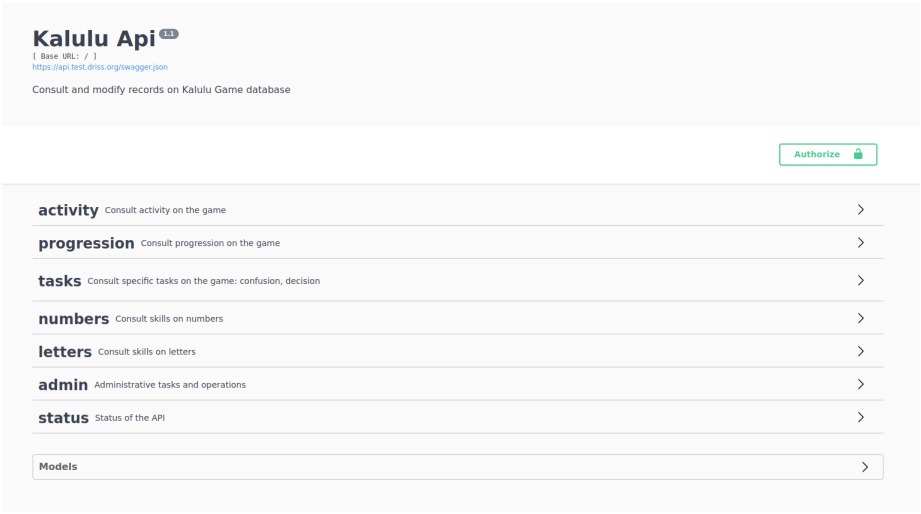


Figure 1: https://research.ludoeducation.fr

Figure 2: Categories endpoint

Category endpoint

All resources that describe the activity of students are stored in a **category endpoint**.

Take a closer look at category called `activity`.

Activity regroups statistics on game usage at student / classroom level consultable on global, on a specific subject or dataset.

This access is usefull for researcher and admin to consult **global activity** of a student or a classroom (timespent, session of gameplay, scores).

> These endpoint offers different level of information to control activity data.

Such endpoint will retrive data that can answer at different level to question such as:

- Is classroom respecting the order of sessions and the frequency?
- What is the average timespent on a subject?
- Is there a difference in timespent and good reponses depending of the order of the session?

- Are student spending more time on chapter session or lesson session?

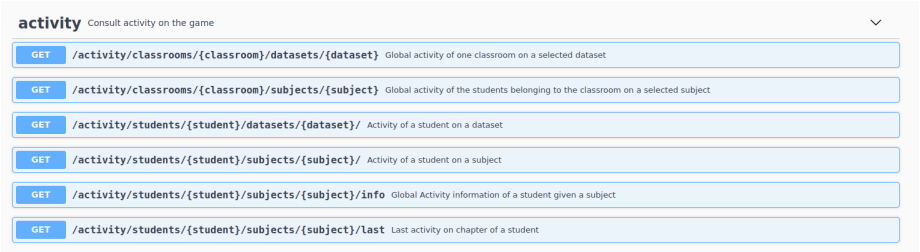
Theses endpoint are also used to display information or feed the plot into the teacher dashboard :

> As an example: Student info is displayed loading data from API endpoint: 'activity/students/1226/subjects/letters/info'

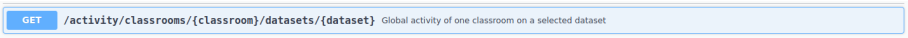
Figure 3: Student Info which data are loaded from endpoint

)

If you click on the category section activity bar: a list of **endpoints** will appear for activity category with a short description.



What is an endpoint API?



API endpoint is an access point to a resource stored in the database. It consists technically of :

- an **URL**: giving the location of the resource (called URI: Unified Resources Identifier) composed of the path and the parameters into `{}`
- a **method**: expressed in an HTTP Verb which corresponds to a specific interaction with the database
  - GET
  - PUT
  - POST

– DELETE Swagger interface added an explicit color code for each of these method.

With those 2 elements, we will perform an HTTP request explicitly describing where is the resource and the interaction with it: simply using the HTTP protocol.

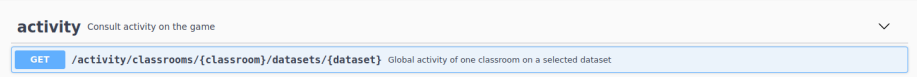


Figure 4: Endpoint example

Endpoint Sections: If you click on one specific endpoint, 4 main sections appears:

- the header endpoint
- the documentation
- the parameters section
- the response section

Header The interface displays for each endpoint:

- the **method** GET in blue to view, PUT in orange to edit/update the resource, DELETE in red to remove the resource, CREATE in green to add a new ressource
- the **url** where the resource is accessible with the **parameters** that are always between `{}`
- a short **description** of the resources availables
- and eventually a **lock** button if the resource is protected throught password

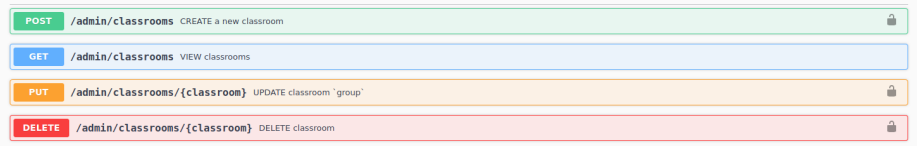


Figure 5: Differents methods are available with color code

Documentation

- Description : explains what kind of information is available
- Methods: give an SQL equivalent of the query made to the database
- Structure of the output (mostly for graphs)
- Link to database documentation

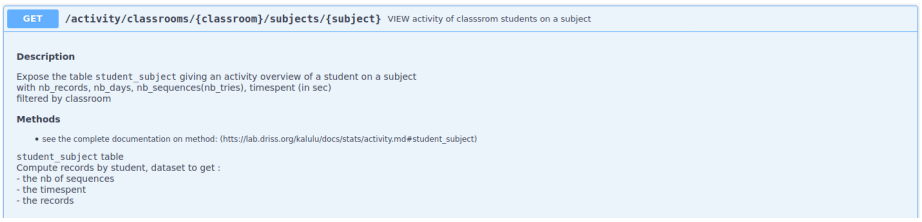


Figure 6: Description and methods use for each endpoint

Parameters Section parameters consists of displaying:

- the name of the different **parameters** availables
- the different **input types** accepted: such as integer, string, coma separated values, string with controled values, ...

Theses parameters always corresponds to elements in brackets inside the url. theses parameters allows to create custom queries using parameters as fields and filters.

Example:

Endoint: `activity/students/{student}/subjects/{subject}/` Query: `activity/students/1226/subjects/letters/info`

Loads activity information on student n° 1226 and on subject letters

try to change the parameters to get other student on other subject

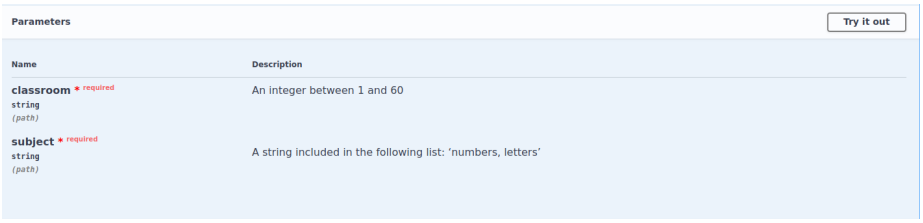


Figure 7: Parameters

Response section

Response section guides you over the responses giving first generic response with code atteched to a textual description that explains the type of error.

Performing a request thought the interface make the response section change: giving more information on the process.

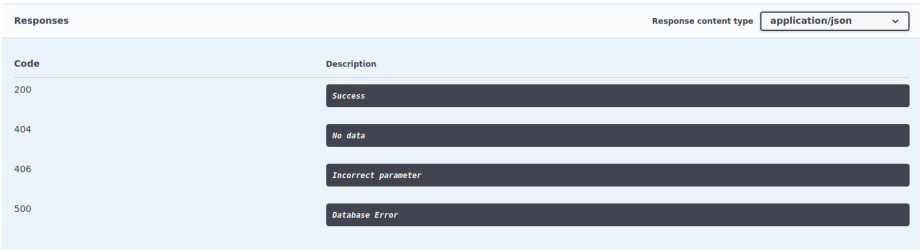


Figure 8: Response

When everything went OK: data is displayed inside the response section with the corresponding 200 code.

When something went wrong a message is return giving contextual information. Depending on the type of error: the code is attached:

- 404 data was not found or resource is not available
- 406 one or multiple parameters were incorrect
- 500 error came from the database or the API (contact administrator in this case)

How to get data from the API?

To access the data you have the differents options:

- Try it online with the web interface
- Try it directly on your web browser
- Load the CSV version of the data
- Use external script to manipulate the data

Try it online

Make the request Press the button `tryit`

Figure 9: Tryit Button

A new form will appear asking to fill out the parameters

Fill the box with the required parameters and press execute

Once executed API shows in the black box the request made

> The tryit online button and forms helps to build the url by filling out parameters Here requests url black box shows

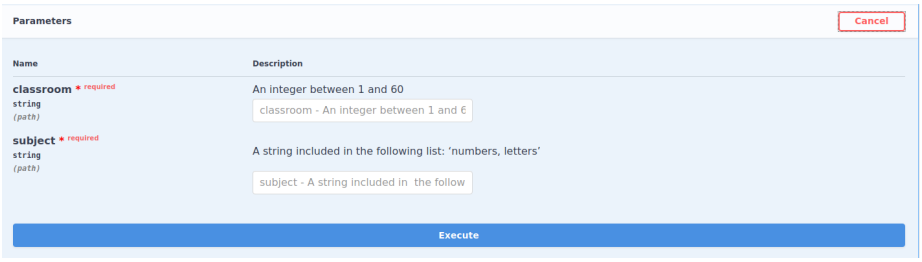


Figure 10: Online form to enter the parameters

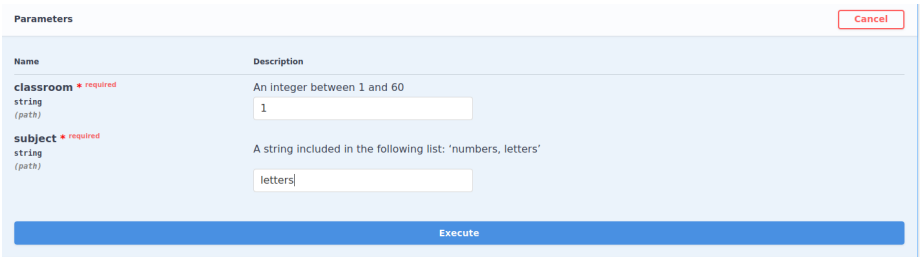


Figure 11: Execution of the form

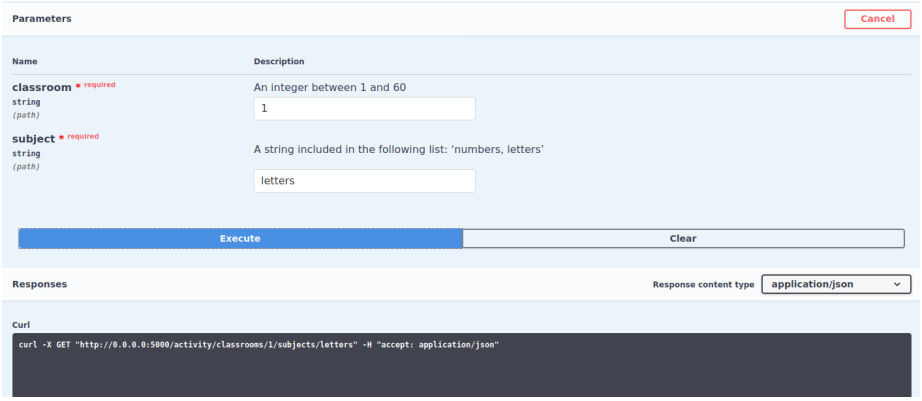


Figure 12: Response to execution

wich request have been made with the correct parameters: for classroom 1 and subject `letters` this only consisting of changing values between the {} in url. For endpoint `API_URL/activity/classrooms/{classroom}/subjects/{subject}` the replacement has been made as the following `API_URL/activity/classrooms/1/subjects/letters`

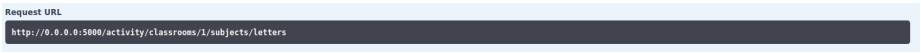


Figure 13: Request url

Understand the response The result is displayed in the section server response: - with a generic **code** and a description **message** - detailled response lays into section `response body`

Detailled type of responses the server can give are listed below and described

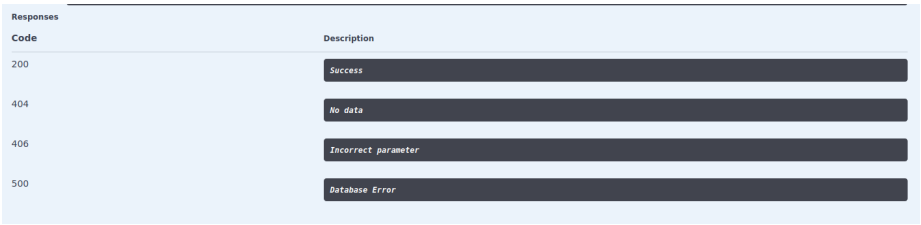


Figure 14: Response types

Response is OK

If response is 200 or 204: no problem have been encountered: API response body displays the data from the database

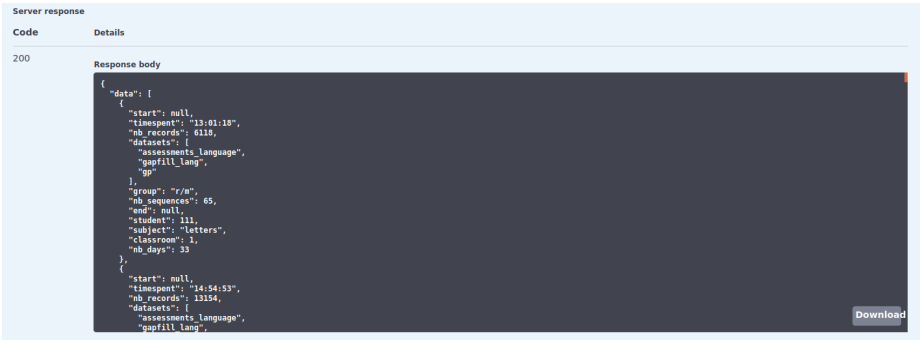


Figure 15: Reponse 200: everything is OK data is displayed

Response is Wrong Each error code (404, 406, 500) corresponds to a certain type of response: a generic error message is attached and detail of the error lays in message body

- If error is 404: no data has been found

Figure 16: Response 4O4: Error resource not found

- If error is 406: parameter given is incorrect. In this case response body message gives the wrong parameter

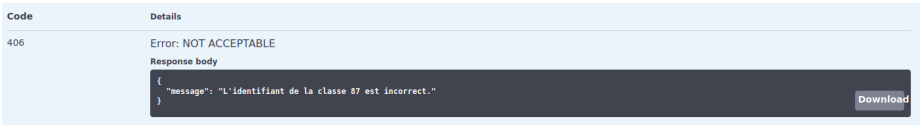


Figure 17: Response 406: Error parameter is incorrect

- If error is 500: table is empty. It means that population of database had a problem and need to be executed again ask for admin to regenerate the database

Direct access

You can also acess it directly into the web browser without loading the interface.

Online tryit button guided you to build the request URL through a web form and helped you to understand the response displaying extended documentation on messages, responses and errors types.

Blue endpoints (GET methods) can be directly accessible into your favorite web browser without filling the form inside the interface as it consists simply on loading a page.

Prepare your request

- In the interface copy the request URL

Change the parameters if you need it

- Paste it into the navigation bar of your favorite web browser
- Press enter

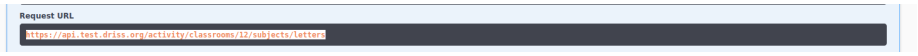


Figure 18: Copy the request inside the section

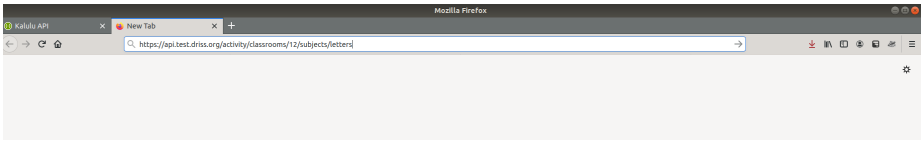


Figure 19: Paste into your navigation bar

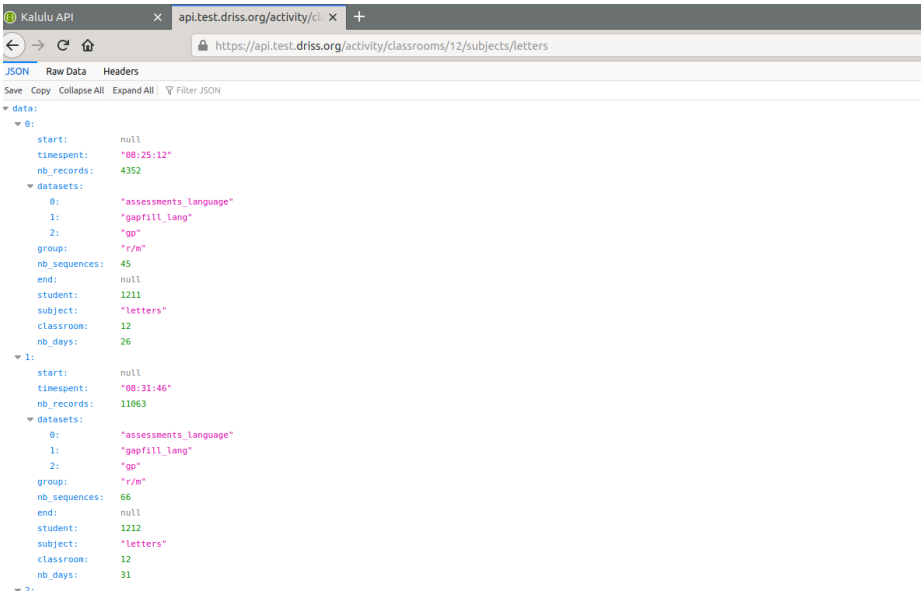


Figure 20: Press enter or click on the arrow

You have now access to the raw response inside your web browser!

Understand the response Status code gives you the information on success/failure of the request

- 200 OK
- 404 Not found
- 406 Wrong parameters

CSV format

Prepare the request

- In the API interface copy the request URL

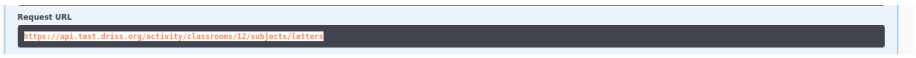


Figure 21: Copy the url in Request url section

- Paste it into the navigation bar of your favorite web browser

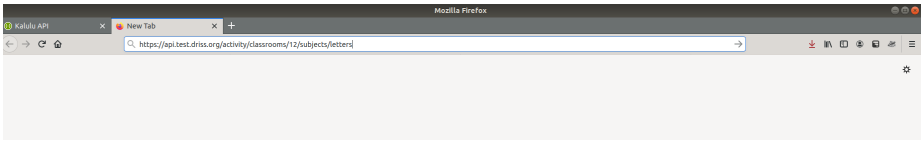


Figure 22: Paste into your navbar

- Add `/csv` at the end of the url

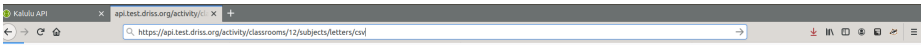


Figure 23: write /csv at the end of the url

- Press enter

You now have access to the data in a CSV format!

Understand the response

Status code inside the HTML page gives you the information on success/failure of the request:

- 200 OK => displays the table on the screen
- 404 Not found => No data
- 406 Wrong parameters => Your request has some problem in parameters

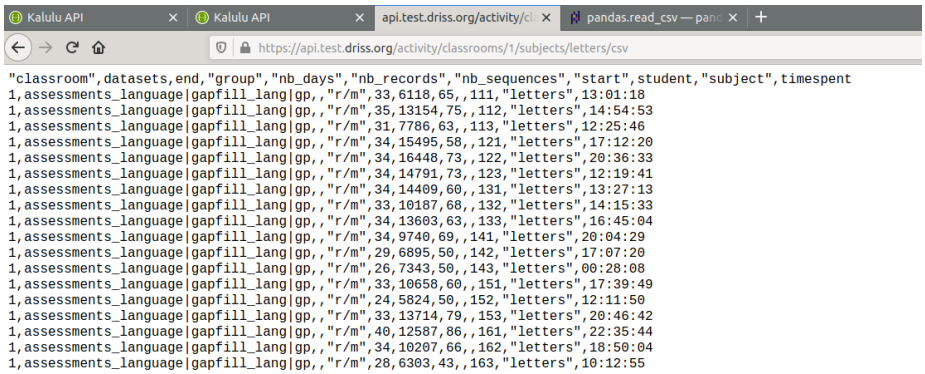


Figure 24: Get the csv

External script

You can access programmatically to the data in CSV and load it.

With python

```
import requests
import pandas
API_URL = "https://www.api.test.driss.org"
#parameters
students = [1223, 1212, 1245]
subjects = ["letters", "numbers"]
for students in students:
    for subject in subjects:
        endpoint = "/activity/students/{}/subjects/{}/csv".format(student,subject)
        r = requests.get(API_URL+endpoint)
        if r.code == 200:
            data = r.text
            csv_data = [x.split(',') for x in data.split('\n')]
            head = csv_data[0]
            df = pd.DataFrame(csv_data[1:], columns=head)
            print(df)
```

With R

An example in R:

```
library (RCurl)
API_URL <- "https://www.api.test.driss.org"
students <- list(1223, 1212, 1245)
```

```
subjects <- list("letters", "numbers")
for(students in students){
    for(subject in subjects){
        endpoint <- str_c(API_URL, "/activity/students/", student, "/subjects/", subject, "/
        html_page <- getURL(endpoint)
    }
}
```

How to update data ?

API provides access points and methods to consult, edit, update, delete specific resources. All theses methods stands in the endpoint category `admin`. Edition, suppression require both an authorisation set throught password to unlock the endpoint access.

The main usage here is to provide facilities to update the stats choosing to integrate or not specific students into the scope of the enquiry.

Main operations: * Unlock access * Consult status of the students * Change group for multiples students * Change group for one classroom * Change group for one student

Unlock access

Some endpoints requires an authorisation as they modify the database and impact the dashboard.

In these case a lock appears on the left side of the endpoint.

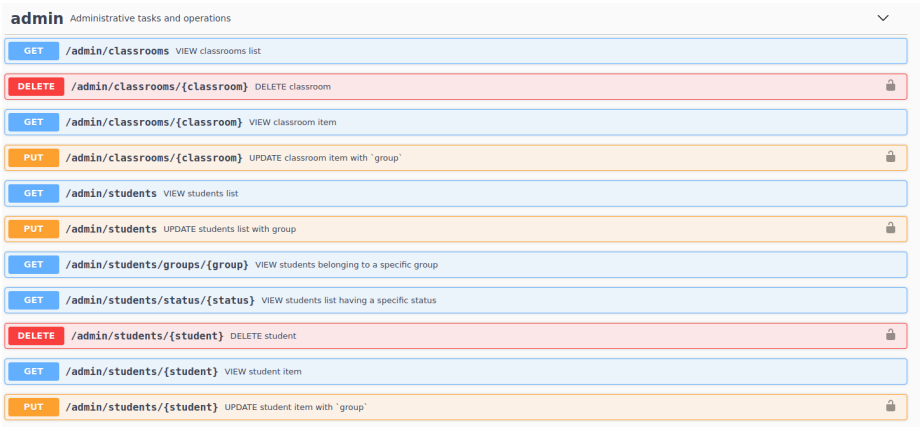


Figure 25: admin_endpoint.png

You can unlock it click on the corresponding icon

You can also unlock using the main button `Authorize` at the top right corner of the interface.

Figure 26: authorize

This authorisation will lasts the time of a session: you don't need to enter the password each time you need to use a locked endpoint.

1. Click on `Authorize` button on the top right corner of the interface.
2. Fill in the form with the password given
3. Click on the `Authorize` button
4. Click on `Close` button (Logout is to revoke the access)

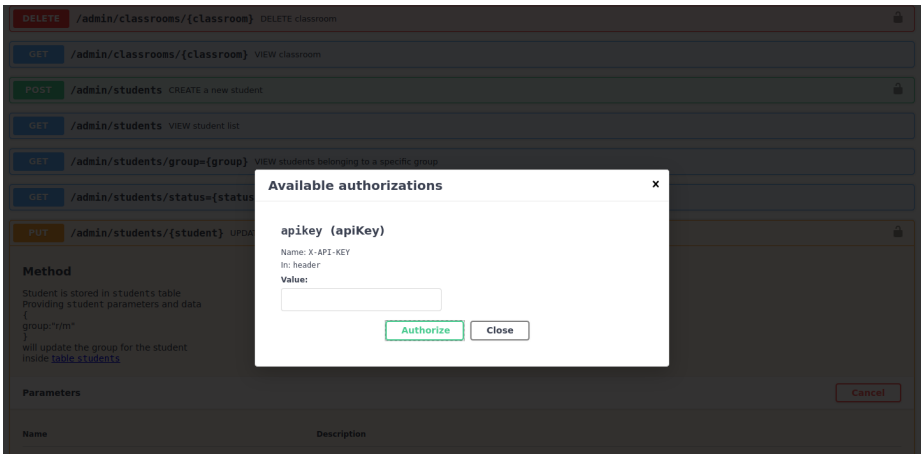
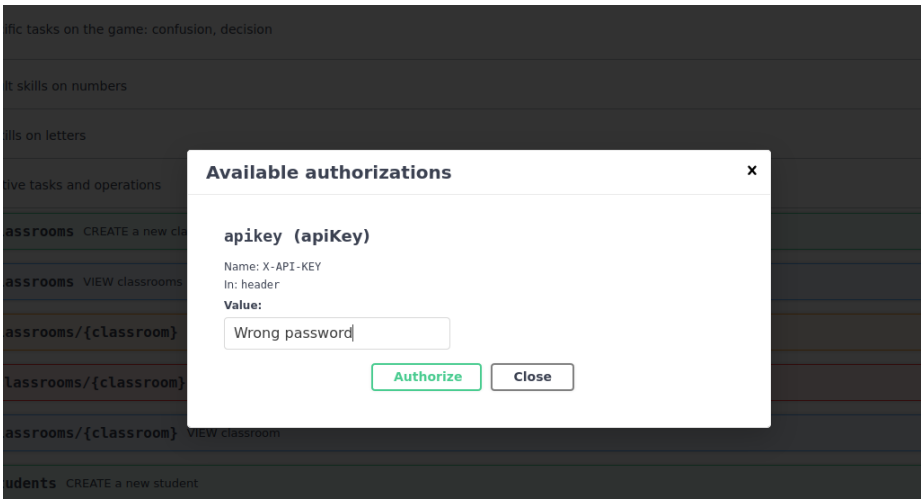
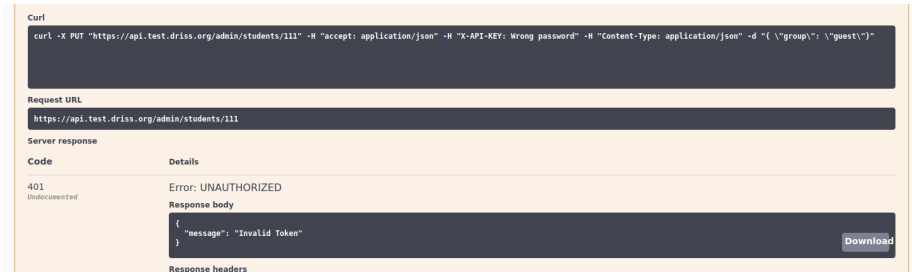


Figure 27: authorize_form.png

- If password is wrong: the request will return a 401 Error UNAUTHORIZED and a message will be displayed {"message": "Token is invalid"}





Consult group of students

- Consult students where group information is None

If group is None: group is missing for the student. It simply means that student was not referenced inside the initial csv listing students with they groups:

in JSON : https://api.test.driss.org/admin/students/groups/None

in CSV : https://api.test.driss.org/admin/students/groups/None/csv

- Consult students which status is False

in JSON : https://api.test.driss.org/admin/students/status/False

in CSV : https://api.test.driss.org/admin/students/status/False/csv

Status of the student can be False for two reasons: - group information is missing for this student (No reference of this student has been found in the initial file that define a group for each student) - student was declared inside the initial file defining the group for each student but no files has been found.

To cover those 2 different cases a specific status is available

- status `missing` means that student was not initially declared but documents have been found so student exists in db but has no group

in JSON : https://api.test.driss.org/admin/students/status/missing

in CSV : https://api.test.driss.org/admin/students/status/missing/csv

- status `empty` means that student was initially declared but no documents have been found. student has only information given by the initial file

in JSON : https://api.test.driss.org/admin/students/status/empty

in CSV : https://api.test.driss.org/admin/students/status/empty/csv

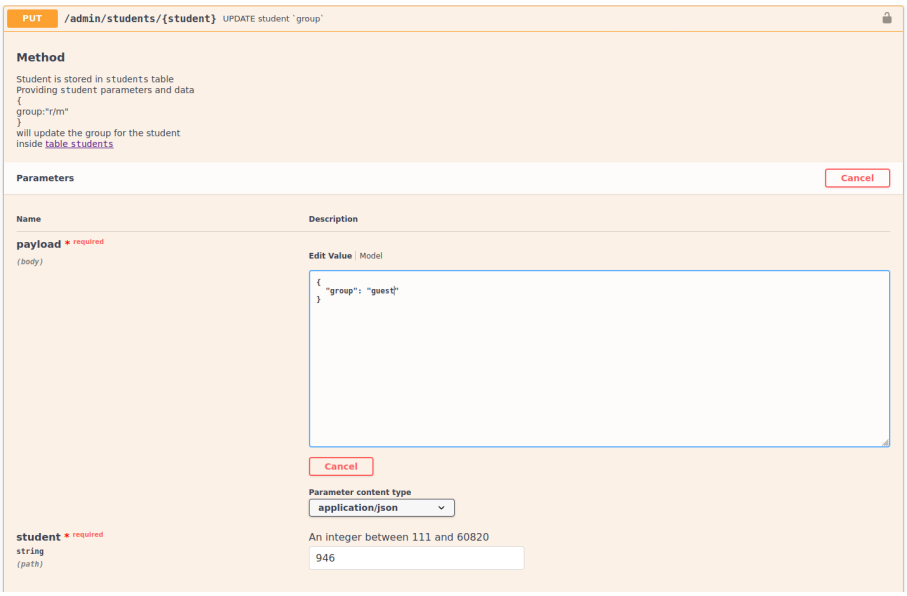
Edit group for one student

You may want to edit a group for a specific student to declare the session he belongs to or to remove him from global stats (this is made by setting the group to `guest`): you will need to use the interface.

Inside admin category endpoint a specific endpoint in orange is available to edit a unique student.

PUT `/admin/students/{student}/`

- Press `try it out` button and the edit from will be shown.
- Fill in the form as following:
  - In parameters enter the student ID e.g 111
  - In payload change the value between quotes attached to group e.g “guest”



- Press execute: > This request will update the group for the given student
  - If everything is OK: API will answer 200 response
  - In case of error: API will display 404, 406 errors and additional information will be displayed in response body

  When group is changed from guest to [“r/m”, “m/r”] this impact the stats: so a background function is fired to add the student to the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed. Be patient!

  When group is changed from [“r/m”, “m/r”] to guest this impact the stats: so a background function is fired to remove the student from the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed. Be patient!

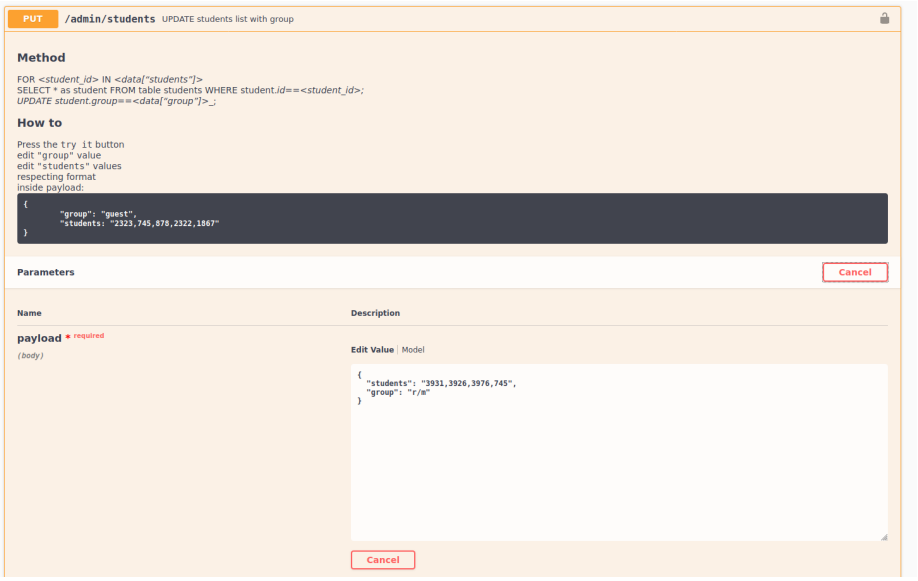
Edit group for muliples student

You may want to edit a group for multiple students that doesn’t necessarily belong to same classroom to declare the session he belongs to or to remove him from global stats (this is made by setting the group to `guest`): you will need to use the interface.

Inside admin category endpoint a specific endpoint in orange is available to edit a multiple student.

PUT `/admin/students/`

- Press `try it out` button and the edit frorm will be shown.



- Fill in the form as following:
  - In payload change the value between quotes attached to group e.g “guest”
  - In payload change the value between quotes attached to students e.g “112, 113, 2987”
- Press execute:

  This request will update the group for all the students in the list

  - If everything is OK: API will answer 200 response
  - In case of error: API will display 404, 406 errors and additional information will be displayed in response body

  When group is changed from guest to [“r/m”, “m/r”] this impact the stats: so a background function is fired to add the student to the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed by student. Be patient!

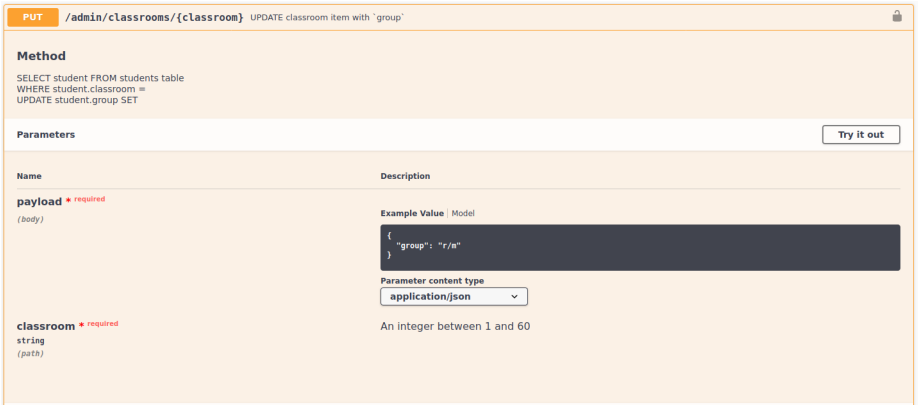
  When group is changed from [“r/m”, “m/r”] to guest this impact the stats: so a background function is fired to remove the student from the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed by student. Be patient!

Edit group for one classroom

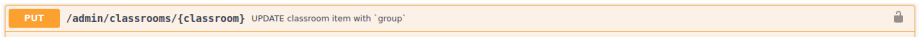
You may want to edit a group for a classroom to declare the session it belongs to or to remove it from global stats (this is made by setting the group to `guest`): you will need to use the interface.

Inside admin category endpoint a specific endpoint in orange is available to edit a multiple student.

PUT `/admin/classrooms/`



- Press `try it out` button and the edit frorm will be shown.



- Fill in the form as following:
  - In parameters enter the classroom ID
  - In payload change the value between quotes attached to group e.g student
- Press execute:

  This request will update the group for all the students belonging to the classroom

  - If everything is OK: API will answer 200 response
  - In case of error: API will display 404, 406 errors
  - additional information will be displayed in response body as a message

  When group is changed from guest to [“r/m”, “m/r”] this impact the stats: so a background function is fired to add the student to the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed by student. Be patient!

  When group is changed from [“r/m”, “m/r”] to guest this impact the stats: so a background function is fired to remove the student from

the stats tables and compute again the global tables. This operation takes about 3 minutes to proceed by student. Be patient!

Receipes

Receipe 1: Edit group for missing students

1. List the student that have no group

GET /admin/students/status/missing/csv

into a variable students store the columns student into a list

and then a string with coma separated values

2. Set all the missing student to 'guest'

PUT /admin/students/

{group: "guest", students:students}

Receipe 2: Consut global activity on a subject for group `r/m`

1. Get the list of all students belonging to group r/m

GET /admin/students/groups/r-m/csv

store list into a variable called students

2. Load stats for each student

subjects = ["numbers", "letters"]

for subject in subjects:

```
for student in students:
```

```
    GET /activity/students/{student}/subject/{subject}
```

Receipe 3: Consut global progression for group 'r/m'

1. Get the list of all students belonging to group r/m

GET /admin/students/groups/r-m/csv

store list into a variable called students

2. Load stats for each students

subjects = ["numbers", "letters"]

for subject in subjects:

```
for student in students:

    GET /progression/chapters/students/{student}/subject/{subject}
```

Receipe 4: Consult global confusion matrix CSV data and graph

GET /tasks/confusion/subject/{subject}/csv

If you want to access the graph confusion matrix on all the student is available on the dashboard at this adress: https://test.driss.org/confusion

Receipe 5: Consult lexical decision for every student of r/m group

1. Get the list of students belonging to group r/m

GET /admin/students/groups/r-m/csv

store list into a variable called students

2. Load stats for students

for student in students:

```
GET /tasks/decision/students/{student}/subjects/letters/csv
```