Documentation API

Table of content

This Project is made with love by Duncan, Iliane and Thibaut Figueira.

Getting started :

The following steps will help you to get started with your journey towards using our beautiful API:

You can chose to use the api on your local pc without any server or use our docker file who contain two server, a web server and a server for the db.

Run the API on your local Pc

1- **First you need to download "Mysql installer", here the link:**

https://dev.mysql.com/downloads/installer/
On the installer, make sure to install "Mysql serveur" and "Mysql workbench".

2- **Configure Mysql workbench:**

Launch "Mysql workbench" and configure the connection with those information:
Server Name = localhost, User = root, password= azerty.
Then on the management section, select import, Import from Self-contains file, select our database on the database folder in the project (mydatabasegood.sql) , then click "New" and name it "newshema". The db should now be displayable on the schemas section.

3- **Launch the Api:**

Now just type :
dotnet run --project serveur-web-GIS/api/
The Api is alive !

4- **Try it on Postman:**

Go on Postman website, and create a account.
Now you can access to our Postman:
https://app.getpostman.com/join-team?invite_code=e33bbf89c4080c120b3fac1a2ec72bf9&target_code=d39e99bc428fa0d95c69a83c7ce353c5
Dont forget to download Postman agent:
https://www.postman.com/downloads/postman-agent/
Try out some request!

Run the API with our Docker

1. **Setup Docker:**

   First, you will need to download Docker desktop on this link:
   https://www.docker.com/products/docker-desktop/

   Don't forget to open it every time you will need to use our api.

2. **Run Docker:**

   In Docker, be sure to delete every existing container, image and volume related to the project before composing.

   Be sure to not have mysqlworkench open when using the api on docker.

   On the terminal, you just have to type:

   docker-compose up --build

   Everything should be good! You just have to test the api with Postman!

3. **Try it on Postman:**

   Go on Postman website, and create a account.
   Now you can access to our Postman:
   https://app.getpostman.com/join-team?invite_code=e33bbf89c4080c120b3fac1a2ec72bf9&target_code=d39e99bc428fa0d95c69a83c7ce353c5
   Dont forget to download Postman agent:
   https://www.postman.com/downloads/postman-agent/
   Try out some request!

Concepts : API calls

Our API is a restful API with different endpoints which return JSON data about users, product, or their shoplist.

Base Url

The base address of our web API is:

http://localhost/api/

Request

Data resources are accessed via standard HTTP requests in UTF-8 format to an API endpoint. The Web API uses the following HTTP verbs:

METHOD	ACTION
GET	Retrieve resources
POST	Creates resources
PUT	Replaces resources or create a new one
PATCH	Partial change of a resources
DELETE	Deletes resources

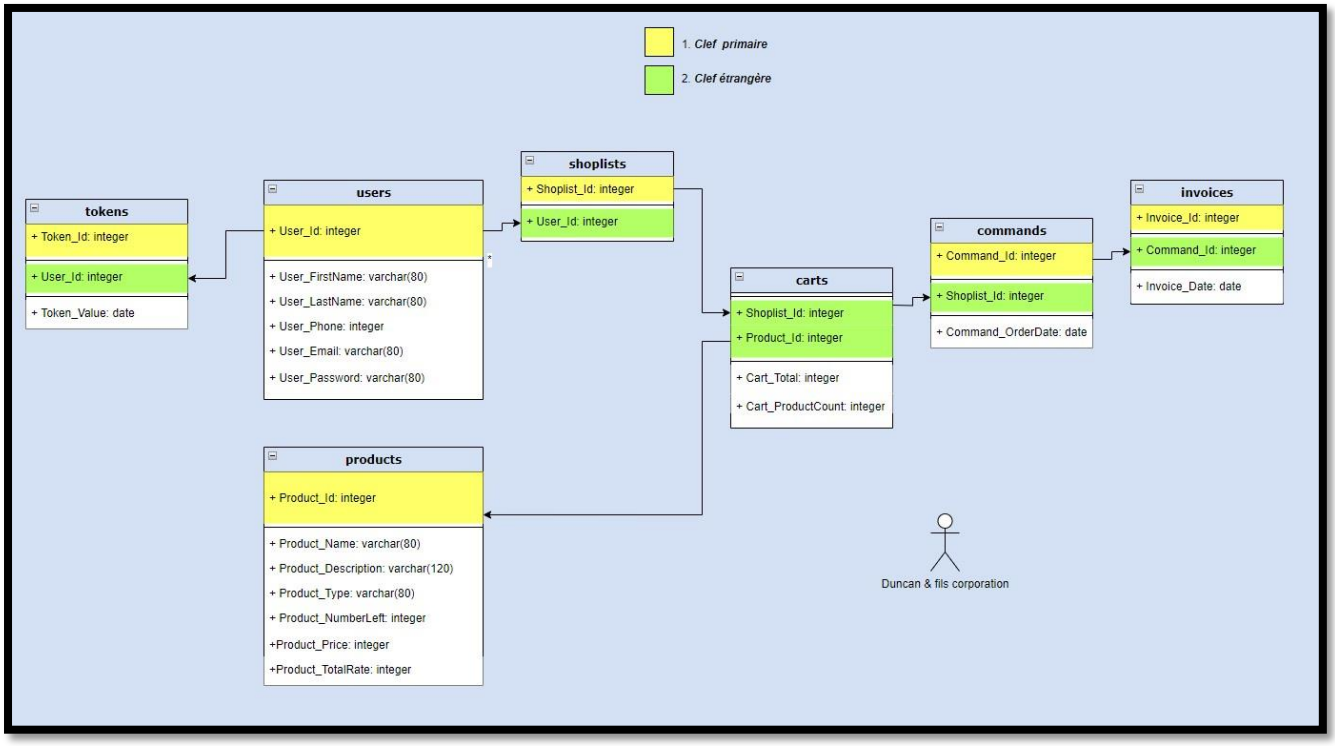
Responses

Web API normally returns JSON in the response body. Some endpoints don't return JSON but the HTTP status code

Response status codes

Status code	Description
200	OK
201	Created
202	Accepted
204	No Content
304	Not Modified
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error
502	Bad Gateway
503	Service Unavailable

DMC



Here the DMC of our DB, all our api endpoint referrer to it.

References endpoint

Running test - Status

Test if the api is running, this is not testing if the db connection work properly.

Method: Get

Endpoint: /api/status

Exemple : *http://localhost:8080/api/status*

Response:

200 – The api is running!

503 – Error, could not send a request. (the api is offline).

Running test – Tokens

Token authorization showcase, gets the user corresponding to the token given.

Method: Get

Endpoint: /api/tokens

Authorization bearer token Required

Bearer Token : admin

Exemple : *http://localhost:8080/api/tokens*

Response:

200 – A Json of the user.

400 – null.

404 – Bad endpoint.

503 – Error, could not send a request. (the api is offline).

Running test – Post a Token

Create a token based on a given body.

Method: Post

Endpoint: /api/tokens

Body Json format Required

```
{
   "User_Id": 5,
   "Token_Value": "adminv2"
}
```

Exemple : *http://localhost:8080/api/tokens*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Users – Get all users

Return in a Json all the users in the db.

Method: Get

Endpoint: /api/users

Exemple : *http://localhost:8080/api/users*

Response:

200 – all the Json users.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get a user by Id

Return in a Json a user corresponding to the passed Id.

Method: Get

Endpoint: /api/users/{myId}

myId int Required

Exemple : *http://localhost:8080/api/users/4*

Response:

200 – A Json of the user.

400 - Invalid id, Error.

400 - Invalid name or email, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get the user shoplist by his Id

Return in a Json a user shoplist corresponding to the passed Id.

Method: Get

Endpoint: /api/users/{myId}/shoplists

myId int Required

Exemple : *http://localhost:8080/api/users/6/shoplists*

Response:

200 – A Json of the user shoplist.

400 - Invalid id or no shoplist found.

404 – Bad endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get the user cart by his Id

Return in a Json a user cart corresponding to the passed Id.

Method: Get

Endpoint: /api/users/{myId}/carts

myId int Required

Exemple : *http://localhost:8080/api/users/5/carts*

Response:

200 – A Json of the user cart.

400 - Invalid id or no cart found.

404 – Bad endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get the user commands by his Id

Return in a Json a user command corresponding to the passed Id.

Method: Get

Endpoint: /api/users/{myId}/commands

myId int Required

Exemple : *http://localhost:8080/api/users/3/commands*

Response:

200 – A Json of the user command.

400 - Invalid id or no command found.

404 – Bad endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get the user invoices by his Id

Return in a Json a user invoice corresponding to the passed Id.

Method: Get

Endpoint: /api/users/{myId}/invoices

myId int Required

Exemple : *http://localhost:8080/api/users/1/invoices*

Response:

200 – A Json of the user invoice.

400 - Invalid id or no invoice found.

404 – Bad endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get a user by his lastname

Return in a Json a user corresponding to the passed lastname.

Method: Get

Endpoint: /api/users/{myLastName}

myLastName string Required

Exemple : *http://localhost:8080/api/users/hessel*

Response:

200 – A Json of the user.

400 - Invalid name or email, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Get a user by his email

Return in a Json a user corresponding to the passed email.

Method: Get

Endpoint: /api/users/{myEmail}

myEmail string Required

Exemple : *http://localhost:8080/api/users/destin96@blick.com*

Response:

200 – A Json of the user.

400 - Invalid name or email, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Users – Post a user

Create a user based on a given body.

Method: Post

Endpoint: /api/users

Body Json format Required

```
{
   "User_FirstName": "Thibaut",
   "User_LastName": "Figueira",
   "User_Email": "thibaut.ynov@hotmail.com",
   "User_Password": "azerty",
   "User_Phone": "0679034009"
}
```

Exemple : *http://localhost:8080/api/users*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Users – Put a user

Remplace all the data of a existing user or if the user doesn't exist, create a user based on a given body and Id.

Method: Put

Endpoint: /api/users/{myId}

myId int Required

Body Json format Required

```
{
   "User_FirstName": "Duncan",
   "User_LastName": "Chaman",
   "User_Email": "Grand.duncan@hotmail.com",
   "User_Password": "azerty",
   "User_Phone": "0890789887"
}
```

Exemple : *http://localhost:8080/api/users/6*

Response:

200 – Its work! User mis a jour!

200 - This Id is empty, New User created.

400 - no or bad body send: {error}.

503 – Error, could not send a request. (the api is offline).

Users – Patch a user

Remplace some data of a existing user based on a given body and Id.

Method: Patch

Endpoint: /api/users/{myId}

myId int Required

Body Json format Required

```
{
   "User_Email": "Iliane.smart@hotmail.fr",
}
```

Exemple : *http://localhost:8080/api/users/7*

Response:

200 – Patch success! User updated!

200 - This Id is empty, New User created.

400 - Bad body.

400 - Invalid id or no rows affected.

503 – Error, could not send a request. (the api is offline).

Users – del a user

Delete a user based on a give Id

Method: Del

Endpoint: /api/users/{myId}

myId int Required

Exemple : *http://localhost:8080/api/users/1*

Response:

200 – Its work! User supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).

Products – Get all products

Return in a Json all the products in the db.

Method: Get

Endpoint: /api/products

Exemple : *http://localhost:8080/api/products*

Response:

200 – all the Json products.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Products – Get a product by Id

Return in a Json a product corresponding to the passed Id.

Method: Get

Endpoint: /api/products/{myId}

myId int Required

Exemple : *http://localhost:8080/api/products/4*

Response:

200 – A Json of the product.

400 - Invalid id, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Products – Get a product by name

Return in a Json a product corresponding to the passed name.

Method: Get

Endpoint: /api/products/{myName}

myName string Required

Exemple : *http://localhost:8080/api/products/beer*

Response:

200 – A Json of the product.

400 - Invalid name, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Products – Get a list of product by type

Return a list of Json product corresponding to the passed type.

Method: Get

Endpoint: /api/products/{myType}

myType string Required

Exemple : *http://localhost:8080/api/products/Peru*

Response:

200 – A Json list of product.

400 - Invalid type, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Products – Post a products

Create a products based on a given body.

Method: Post

Endpoint: /api/products

Body Json format Required

```
{
   "Product_Name": "bbb",
   "Product_Description": "aaa",
   "Product_Type": "eee",
   "Product_NumberLeft": 9,
   "Product_Price": 12
}
```

Exemple : *http://localhost:8080/api/products*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Products – Put a product

Remplace all the data of a existing product or if the product doesn't exist, create a product based on a given body and Id.

Method: Put

Endpoint: /api/products/{myId}

myId int Required

Body Json format Required

```
{
   "Product_Name": "test",
   "Product_Description": "yoyooooooooooooooooooooooooooooooooooooo",
   "Product_Type": "beer",
   "Product_NumberLeft":7,
   "Product_Price": 4
}
```

Exemple : *http://localhost:8080/api/products/2*

Response:

200 – Its work! Product mis a jour!

200 - This Id is empty, New Product created.

400 - no or bad body send: {error}.

503 – Error, could not send a request. (the api is offline).

Products – Patch a product

Remplace some data of a existing product based on a given body and Id.

Method: Patch

Endpoint: /api/product/{myId}

myId int Required

Body Json format Required

```
{
   "Product_Description": "uwu",
}
```

Exemple : *http://localhost:8080/api/products/3*

Response:

200 – Patch success! Product updated!

200 - This Id is empty, New Product created.

400 - Bad body.

400 - Invalid id or no rows affected.

503 – Error, could not send a request. (the api is offline).

Products – Del a product

Delete a product based on a give Id

Method: Del

Endpoint: /api/products/{myId}

myId int Required

Exemple : *http://localhost:8080/api/products/4*

Response:

200 – Its work! Product supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).

Shoplists – Get all shoplist

Return in a Json all the shoplist in the db.

Method: Get

Endpoint: /api/shoplists

Exemple : *http://localhost:8080/api/shoplists*

Response:

200 – All the Json shoplist.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Shoplists – Get a shoplist by Id

Return in a Json a shoplist corresponding to the passed Id.

Method: Get

Endpoint: /api/shoplists/{myId}

myId int Required

Exemple : *http://localhost:8080/api/shoplists/4*

Response:

200 – A Json of the shoplist.

400 - Invalid id, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Shoplists – Post a shoplist

Create a shoplist based on a given body.

Method: Post

Endpoint: /api/shoplists

Body Json format Required

```
{
   "User_Id": 7
}
```

Exemple : *http://localhost:8080/api/shoplists*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Shoplists – Put a shoplist

Remplace all the data of a existing shoplist or if the shoplist doesn't exist, create a shoplist based on a given body and Id.

Method: Put

Endpoint: /api/shoplists/{myId}

myId int Required

Body Json format Required

```
{
   "User_Id": 9
}
```

Exemple : *http://localhost:8080/api/shoplists/7*

Response:

200 – Its work! shoplist mis a jour!

200 - This Id is empty, new shoplist created.

400 - no or bad body send: {error}.

503 – Error, could not send a request. (the api is offline).

Shoplists – Patch a shoplists

Remplace some data of a existing shoplist based on a given body and Id.

Method: Patch

Endpoint: /api/shoplists/{myId}

myId int Required

Body Json format Required

```
{
   "User_Id": 7
}
```

Exemple : *http://localhost:8080/api/shoplists/8*

Response:

200 – Patch success! Shoplist updated!

200 - This Id is empty, New Shoplist created.

400 - Bad body.

400 - Invalid id or no rows affected.

503 – Error, could not send a request. (the api is offline).

Shoplists – Del a shoplist

Delete a shoplist based on a give Id

Method: Del

Endpoint: /api/shoplists/{myId}

myId int Required

Exemple : *http://localhost:8080/api/shoplists/9*

Response:

200 – Its work! Shoplist supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).

Carts – Get all cart

Return in a Json all the cart in the db.

Method: Get

Endpoint: /api/carts

Exemple : *http://localhost:8080/api/carts*

Response:

200 – All the Json cart.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Carts – Get a cart by Id

Return in a Json a cart corresponding to the passed Id.

Method: Get

Endpoint: /api/carts/{myId}

myId int Required

Exemple : *http://localhost:8080/api/carts/4*

Response:

200 – A Json of the cart.

400 - Invalid id, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Carts – Post a cart

Create a cart based on a given body.

Method: Post

Endpoint: /api/carts

Body Json format Required

```
{
   "Shoplist_Id": 7,
   "Product_Id": 7,
   "Cart_Total": 85,
   "Cart_ProductCount": 5
}
```

Exemple : *http://localhost:8080/api/carts*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Carts – Del a cart

Delete a cart based on a give Id

Method: Del

Endpoint: /api/carts/{myId}

myId int Required

Exemple : *http://localhost:8080/api/carts/7*

Response:

200 – Its work! Cart supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).

Commands – Get all command

Return in a Json all the command in the db.

Method: Get

Endpoint: /api/commands

Exemple : *http://localhost:8080/api/commands*

Response:

200 – All the Json command.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Commands – Get a command by Id

Return in a Json a command corresponding to the passed Id.

Method: Get

Endpoint: /api/commands/{myId}

myId int Required

Exemple : *http://localhost:8080/api/commands/4*

Response:

200 – A Json of the command.

400 - Invalid id, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Commands– Post a command

Create a command based on a given body.

Method: Post

Endpoint: /api/commands

Body Json format Required

```
{
   "Shoplist_Id": 9,
   "Command_OrderDate": "2011-05-21"
}
```

Exemple : *http://localhost:8080/api/commands*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Commands – Del a command

Delete a command based on a give Id

Method: Del

Endpoint: /api/commands/{myId}

myId int Required

Exemple : *http://localhost:8080/api/commands/7*

Response:

200 – Its work! Command supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).

Invoices – Get all invoice

Return in a Json all the invoice in the db.

Method: Get

Endpoint: /api/invoices

Exemple : *http://localhost:8080/api/invoices*

Response:

200 – All the Json invoice.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Invoices – Get a invoice by Id

Return in a Json a invoice corresponding to the passed Id.

Method: Get

Endpoint: /api/invoices/{myId}

myId int Required

Exemple : *http://localhost:8080/api/invoices/4*

Response:

200 – A Json of the invoice.

400 - Invalid id, Error.

404 – Invalid endpoint.

503 – Error, could not send a request. (the api is offline).

Invoices– Post a invoice

Create a invoice based on a given body.

Method: Post

Endpoint: /api/invoices

Body Json format Required

```
{
  "Command_Id": 8,
  "Invoice_Date": "2011-05-21"
}
```

Exemple : *http://localhost:8080/api/invoices*

Response:

200 – Its work! Post effectué!

400 - Error during post: {error}.

400 – Invalid Json Format.

503 – Error, could not send a request. (the api is offline).

Invoices – Del a invoice

Delete a invoice based on a give Id

Method: Del

Endpoint: /api/invoices/{myId}

myId int Required

Exemple : *http://localhost:8080/api/invoices/7*

Response:

200 – Its work! Invoice supprimer!

400 - Invalid id.

503 – Error, could not send a request. (the api is offline).