

Projet : API POKÉMON

CRÉATION API REST

Jénitat INDIANA -B3-DEVOPS

Sommaire

[Introduction : 4](#_Toc126364249)

[Sujet : 4](#_Toc126364250)

[Technologies utilisées 5](#_Toc126364251)

[**MySQL Workbench :** 5](#_Toc126364252)

[**MySQL Community Server :** 5](#_Toc126364253)

[**Insomnia :** 5](#_Toc126364254)

[**Visual Studio Code :** 5](#_Toc126364255)

[Solution 5](#_Toc126364256)

[**Logiciel installer et outils :** 5](#_Toc126364257)

[**Librairies installer :** 5](#_Toc126364258)

[**Lancement de l’application :** 6](#_Toc126364259)

[**Création de la base de données :** 7](#_Toc126364260)

[**Les Requette : Insomnia :** 11](#_Toc126364261)

[1. http://127.0.0.1:8000/pokemons : **Récupère la liste de tous les Pokémons** 11](#_Toc126364262)

[2. http://127.0.0.1:8000/pokemons/id/1 **Récupère les détails du pokémon précisé par :id** 12](#_Toc126364263)

[3. http://127.0.0.1:8000/pokemons/name/Charmander **: Récupère les détails du pokémon précisé par :name** 12](#_Toc126364264)

[4. http://127.0.0.1:8000/types/id/2 **Récupère les détails du type précisé par :id** 13](#_Toc126364265)

[5. http://127.0.0.1:8000/abilities **Récupère la liste de toutes les compétences** 13](#_Toc126364266)

[6. http://127.0.0.1:8000/abilities/1 **Récupère les détails de la compétence précisée par :id** 14](#_Toc126364267)

[7. http://127.0.0.1:8000/pokemons/id/types/2 **Récupère le ou les types du pokémon précisé par :id** 15](#_Toc126364268)

[8. http://127.0.0.1:8000/pokemons/id/abilities/2 **Récupère les compétences apprenables du pokémon précisé par :id** 15](#_Toc126364269)

[9. **Ajout d’un pokémon (À noter pour exécuter les commandes de 9 à 11, il faut les exécuter dans l’ordre 11, 10 et puis 9 à cause des clés étrangères qui relient les 3 tables ensembles)** 16](#_Toc126364270)

[Base de données avant d’ajouter le nouveau pokémon : 16](#_Toc126364271)

[On ajoute un nouveau pokémon en utilisant le lien http://127.0.0.1:8000/pokemons/ 16](#_Toc126364272)

[Base de données après d’avoir rajouter le pokémon : 17](#_Toc126364273)

[10. **Ajout d’une compétence** 18](#_Toc126364274)

[Base de données avant de rajouter la nouvelle compétence du pokémon : 18](#_Toc126364275)

[On rajoute une nouvelle compétence en utilisant le lien http://127.0.0.1:8000/abilities/ 18](#_Toc126364276)

[Base de données après avoir rajouter la nouvelle compétence : 19](#_Toc126364277)

[11. **Ajout d’un type** 20](#_Toc126364278)

[Base de données avant de rajouter le nouveau type. 20](#_Toc126364279)

[On rajoute un nouveau type avec notre API. On utilise le lien http://127.0.0.1:8000/type/ 20](#_Toc126364280)

[Base de données après avoir rajouter le nouveau type : 21](#_Toc126364281)

[12. **Modification du pokémon précisé par :id** 22](#_Toc126364282)

[Base de données avant modification : 22](#_Toc126364283)

[Pour modifier les informations du pokémon on utilise le lien : http://127.0.0.1:8000/pokemons/1 22](#_Toc126364284)

[Base de données après modification : 23](#_Toc126364285)

[13. **Modification de la compétence précisée par :id** 24](#_Toc126364286)

[Base de données avant modification : 24](#_Toc126364287)

[Pour modifier la compétence on utilise le lien http://127.0.0.1:8000/abilities/1 24](#_Toc126364288)

[Base de données après modification : 25](#_Toc126364289)

[14. **Modification du type précisé par :id** 26](#_Toc126364290)

[Base de données avant modification : 26](#_Toc126364291)

[Pour modifier le type on utilise le lien http://127.0.0.1:8000/type/1 26](#_Toc126364292)

[La base de données après modification du type : 27](#_Toc126364293)

[15. **Suppression du pokémon précisé par :id** 28](#_Toc126364294)

[Base de données avant suppression : 28](#_Toc126364295)

[On supprime un pokémon en utilisant ce lien http://127.0.0.1:8000/pokemons/7 28](#_Toc126364296)

[Base de données après suppression du pokémon : 29](#_Toc126364297)

[16. **Suppression de la compétence précisée par :id** 29](#_Toc126364298)

[Base de données avant suppression de la compétence : 29](#_Toc126364299)

[On supprime la compétence en utilisant le lien http://127.0.0.1:8000/abilities/7 29](#_Toc126364300)

[Base de données après qu’on a supprimé la compétence : 30](#_Toc126364301)

[17. **Suppression du type précisé par :id** 31](#_Toc126364302)

[Base de données avant de supprimer le type : 31](#_Toc126364303)

[Pour supprimer un type, on utilise le lien : http://127.0.0.1:8000/type/7 32](#_Toc126364304)

[Base de données après avoir supprimer le type : 32](#_Toc126364305)

[Conclusion 34](#_Toc126364306)

[Bibliographie 35](#_Toc126364307)

# Introduction :

L'API REST est l'interface de programmation d’applications de transfert d'état représentatives, habituellement aussi connu sous le nom de REST API Web Service. C'est-à-dire, lorsque l'API RESTful est appelée, le serveur transfère pour représenter l'état des ressources demandées au système client.

Il est également défini comme un style architectural et une méthodologie couramment utilisés dans le développement des services Internet, tels que les systèmes hypermédia distribués.

Ce fonctionnement, le REST détermine la structure d’un API. L'API REST décompose une transaction pour générer une séquence de petits composants, chaque composant répond à un aspect fondamental particulier d'une transaction. Cette modularité la rend flexible sur le plan du développement.

Il exploite les méthodes http décrites par le protocole RFC 2016. Il utilise les requêtes GET économisez nécessaire récupérer des données, PUT nécessaire pour modifier l’état des données, POST demande pour créer des données, DELETE pour supprimer la demande pour l’éliminer.

# Sujet :

Création de API REST qui exposant les requêtes suivantes :

1. GET - /api/pokemons : Récupère la liste de tous les pokémons
2. GET - /api/pokemons/ : id : Récupère les détails du pokémon précisé par :id
3. GET - /api/pokemons/ : name : Récupère les détails du pokémon précisé par :name
4. GET - /api/types/ : id : Récupère les détails du type précisé par :id
5. GET - /api/abilities : Récupère la liste de toutes les compétences
6. GET - /api/abilities/ :id : Récupère les détails de la compétence précisée par :id
7. GET - /api/pokemons/ :id/types : Récupère-le ou les types du pokémon précisé par :id
8. GET - /api/pokemons/ :id/abilities : Récupère les compétences apprenables du pokémon précisé par :id
9. POST - /api/pokemons : Ajout d’un pokémon
10. POST - /api/abilities : Ajout d’une compétence
11. POST - /api/types : Ajout d’un type
12. PUT - /api/pokemons/ :id : Modification du pokémon précisé par :id
13. PUT - /api/abilities/ :id : Modification de la compétence précisée par :id
14. PUT - /api/type/ :id : Modification du type précisé par :id
15. DELETE - /api/pokemons/ :id : Suppression du pokémon précisé par :id
16. DELETE - /api/abilities/ :id : Suppression de la compétence précisée par :id
17. DELETE - /api/type/ :id : Suppression du type précisé par :id

# Technologies utilisées

## **MySQL Workbench :**

C’est un outil visuel unifié pour les architectes de bases de données. Il fournit la modélisation des données, le développement SQL et des outils d'administration complets pour la configuration du serveur, l'administration des utilisateurs, la sauvegarde et bien plus encore.

## **MySQL Community Server :**

Est une édition offre les fonctions basiques d’un système de gestion de base de données, comme la création de tables, les views, les triggers et les procédures stockées.

## **Insomnia :**

Le logiciel Insomnia est un “client HTTP“, c'est-à-dire un programme qui permet d'effectuer des requêtes HTTP sur n'importe quel serveur web, en local ou distant.

## **Visual Studio Code :**

Est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS.

# Solution

## **Logiciel installer et outils :**

* **MySQL Workbench :** pour la création de la base de données et la manipulation de données dans un premier temps.
* **MySQL Community Server :** pour avoir un serveur de base de données.
* **Insomnia :** pour tester les requêtes de l’API, FastAPI.
* **Python :** langage de programmation pour utiliser FastAPI et aussi pour envoyer des requêtes à notre base de données.

## **Librairies installer :**

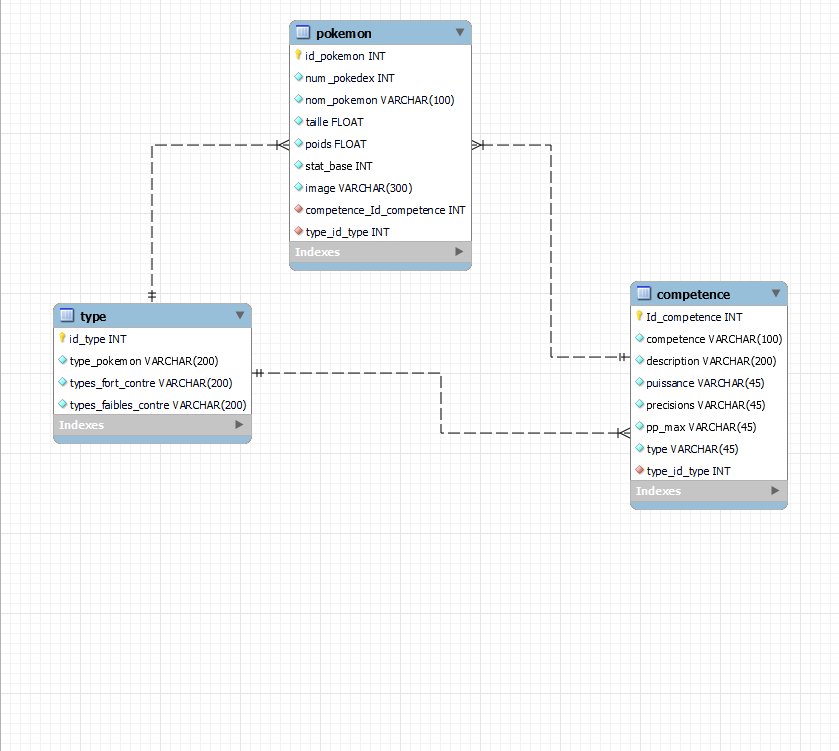
Ci-dessous les différentes librairies qu’il faut installer pour pouvoir coder l’API. Dans une invite de commande typer les lignes ci-dessous :

* **pip3 install fastapi --user**
* **pip3 install uvicorn --user**
* **pip3 install pydantic –user**
* **pip3 install mysql-connector-python**
* **pip installe** par défaut les packages Python dans un répertoire système (tel que /usr/local/lib/python3.4). Cela nécessite un accès root.
* C’est pour cela on utilise **--user** pour créer des packages d'installation pip dans le répertoire personnel à la place, ce qui ne nécessite aucun privilège spécial.
* **FastAPI** est une librairie avec laquelle on créer notre propre API.
* **Uvicorn** est un serveur Web compatible ASGI (async server gateway interface). Ça simplifie l'élément de liaison qui gère les connexions Web à partir du navigateur ou du client API, puis permet à FastAPI de répondre à la demande réelle.
* **mysql-connector-python** comme le nom l’indique, on va utiliser cette librairie pour nous connecter à notre base de données.
* **Pydantic** est une bibliothèque utile pour l'analyse et la validation des données. Il contraint les types d'entrée au type déclaré. On va utiliser cette librairie pour créer des modèles avec pour manipuler les informations depuis les requêtes de type POST ou PUT.

## **Lancement de l’application :**

* Tout d’abord on lance MySQL Workbench avec privilège administratif pour se connecter à notre base de données.
* Dans le répertoire de notre projet, avec une invite de commande on lance uvicorn avec cette ligne, uvicorn main:app --reload. Le main c’est le nom de notre fichier python, app c’est le nom qu’on a donné à notre API et le –reload est utilisé pour relancer notre server uvicorn à chaque fois qu’il y a un changement dans notre code de python.
* Puis on lance Insomnia pour tester les requêtes de notre API.

## **Création de la base de données :**



**Remarque :**

Dans la table type et competence, il y avait la colonne type en commun, et quand je tester pour récupérer les informations, l’API retourner qu’une seule valeur des 2 colonnes type. Alors j’ai dû renommer la colonne type dans type a type pokémon pour que toutes les informations nécessaires soit remonter dans l’API.

Dans la table pokémon j’ai renommé la colonne nom à pokémon car dans certaines requêtes de l’API, les informations stocker dans ce colonne-là ne s’afficher pas à l’ecran.

-- MySQL Script generated by MySQL Workbench

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `pokemon` DEFAULT CHARACTER SET utf8 ;

USE `pokemon` ;

-- -----------------------------------------------------

-- Table `mydb`.`type`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `pokemon`.`type` (

`id\_type` INT NOT NULL,

`type\_pokemon` VARCHAR(200) NOT NULL,

`types\_fort\_contre` VARCHAR(200) NOT NULL,

`types\_faibles\_contre` VARCHAR(200) NOT NULL,

PRIMARY KEY (`id\_type`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`competence`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `pokemon`.`competence` (

`Id\_competence` INT NOT NULL,

`competence` VARCHAR(100) NOT NULL,

`description` VARCHAR(200) NOT NULL,

`puissance` VARCHAR(45) NOT NULL,

`precisions` VARCHAR(45) NOT NULL,

`pp\_max` VARCHAR(45) NOT NULL,

`type` VARCHAR(45) NOT NULL,

`type\_id\_type` INT NOT NULL,

PRIMARY KEY (`Id\_competence`),

INDEX `fk\_competence\_type1\_idx` (`type\_id\_type` ASC) VISIBLE,

CONSTRAINT `fk\_competence\_type1`

FOREIGN KEY (`type\_id\_type`)

REFERENCES `pokemon`.`type` (`id\_type`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`pokemon`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `pokemon`.`pokemon` (

`id\_pokemon` INT NOT NULL,

`num\_pokedex` INT NOT NULL,

`nom\_pokemon` VARCHAR(100) NOT NULL,

`taille` FLOAT NOT NULL,

`poids` FLOAT NOT NULL,

`stat\_base` INT NOT NULL,

`image` VARCHAR(300) NOT NULL,

`competence\_Id\_competence` INT NOT NULL,

`type\_id\_type` INT NOT NULL,

INDEX `fk\_pokemon\_competence\_idx` (`competence\_Id\_competence` ASC) VISIBLE,

INDEX `fk\_pokemon\_type1\_idx` (`type\_id\_type` ASC) VISIBLE,

PRIMARY KEY (`id\_pokemon`),

CONSTRAINT `fk\_pokemon\_competence`

FOREIGN KEY (`competence\_Id\_competence`)

REFERENCES `pokemon`.`competence` (`Id\_competence`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_pokemon\_type1`

FOREIGN KEY (`type\_id\_type`)

REFERENCES `pokemon`.`type` (`id\_type`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

insert into type values

(1, 'Herbe, Poison', 'Herbe, Eau, Electrique, Combat, Fee', 'Air, Feu Glace, Psychique'),

(2, 'Herbe, Poison', 'Herbe, Eau, Electrique, Combat, Fee', 'Air, Feu Glace, Psychique'),

(3, 'Herbe, Poison', 'Herbe, Eau, Electrique, Combat, Fee', 'Air, Feu Glace, Psychique'),

(4, 'Feu','Insecte, Acier, Feu, Herbe, Glace, Fee', 'Terre, Roche, Eau'),

(5, 'Feu','Insecte, Acier, Feu, Herbe, Glace, Fee', 'Terre, Roche, Eau'),

(6, 'Feu, Air','Terre, Insecte, Acier, Feu, Herbe, Glace, Fee, Combat', 'Electrique, Roche, Eau')

insert into competence values

(1, 'Proliferation', 'Augmentez les mouvements de type Herbe lorsque les PV du Pokémon sont faibles.','49','100%', '35', 'Herbe', 1),

(2, 'Proliferation', 'Augmentez les mouvements de type Herbe lorsque les PV du Pokémon sont faibles.','49','100%', '35', 'Herbe', 2),

(3, 'Proliferation', 'Augmentez les mouvements de type Herbe lorsque les PV du Pokémon sont faibles.','49','100%', '35', 'Herbe', 3),

(4, 'Flamber', 'Augmentez les mouvements de type Feu lorsque les PV du Pokémon sont faibles.', '40', '100%', '40', 'Feu', 4),

(5, 'Flamber', 'Augmentez les mouvements de type Feu lorsque les PV du Pokémon sont faibles.', '40', '100%', '40', 'Feu', 5),

(6, 'Flamber', 'Augmentez les mouvements de type Feu lorsque les PV du Pokémon sont faibles.', '40', '100%', '40', 'Feu', 6)

insert into pokemon values

(1, 1, 'Bulbasaur', 0.7, 6.9, 64,'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/1.png',1 ,1),

(2, 2, 'Ivysaur', 1, 13, 142, 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/2.png' ,2 ,2),

(3, 3, 'Venusaur', 2, 100, 208,'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/3.png' ,3 ,3),

(4, 4, 'Charmander', 0.6, 8.5, 65, 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/4.png', 4, 4),

(5, 5, 'Charmaleon', 1.1, 19, 142, 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/5.png', 5, 5),

(6, 6, 'Charizard', 1.7, 90.5, 209, 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/6.png', 6, 6)

## **Les Requette : Insomnia :**

## <http://127.0.0.1:8000/pokemons> : **Récupère la liste de tous les Pokémons**

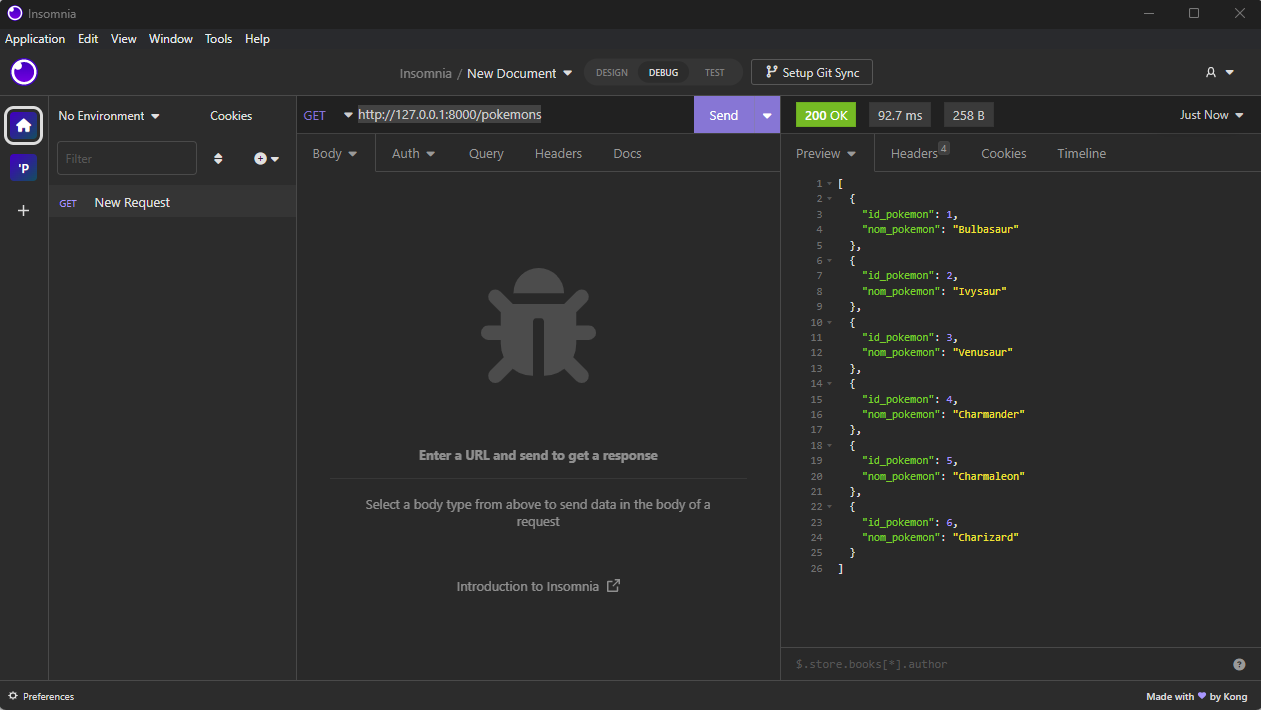


Figure Récupère la liste de tous les Pokémons

## <http://127.0.0.1:8000/pokemons/id/1> **Récupère les détails du pokémon précisé par :id**

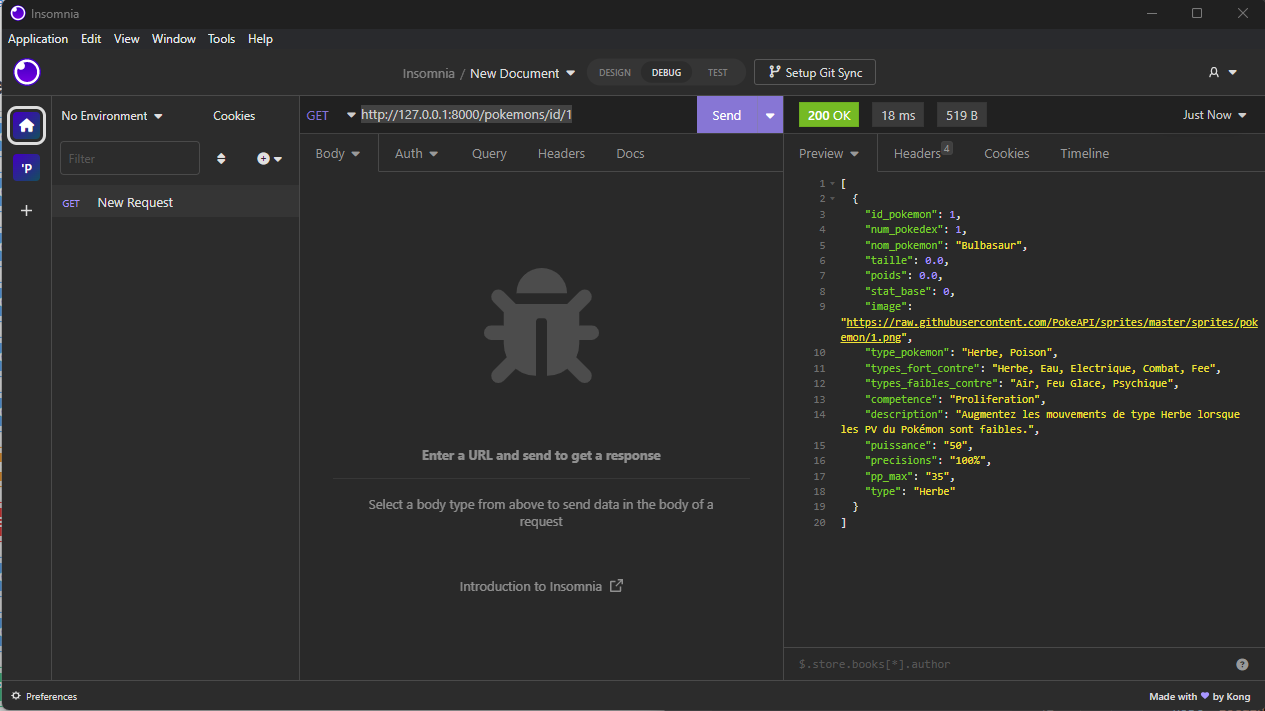


Figure Récupère les détails du pokémon précisé par : id

## <http://127.0.0.1:8000/pokemons/name/Charmander> **: Récupère les détails du pokémon précisé par :name**

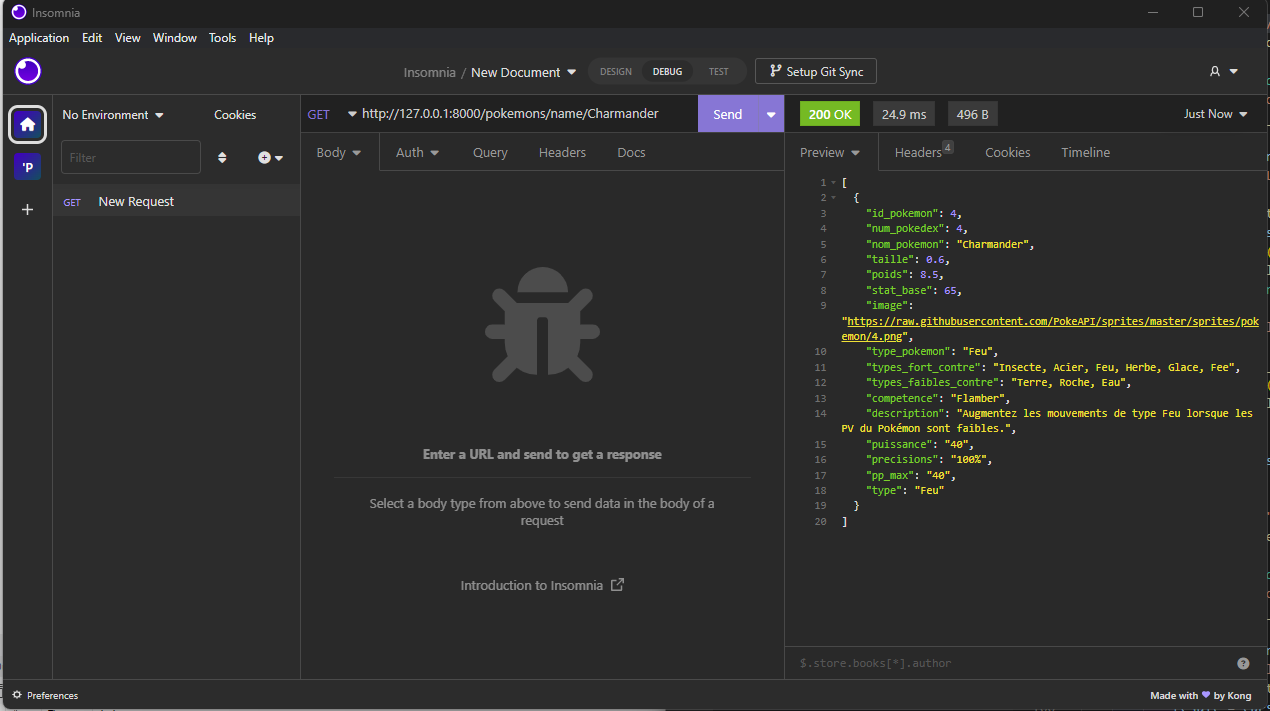


Figure Récupère les détails du pokémon précisé par :name

## <http://127.0.0.1:8000/types/id/2> **Récupère les détails du type précisé par :id**

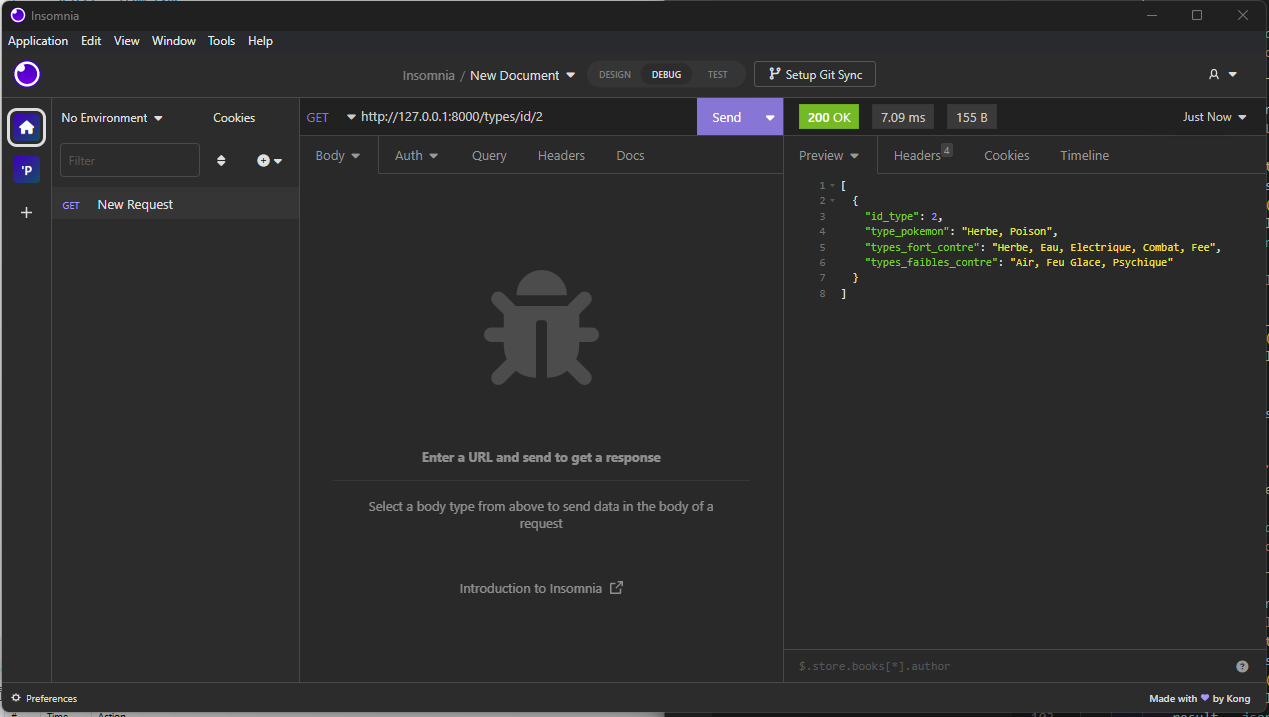


Figure Récupère les détails du type précisé par :id

## <http://127.0.0.1:8000/abilities> **Récupère la liste de toutes les compétences**

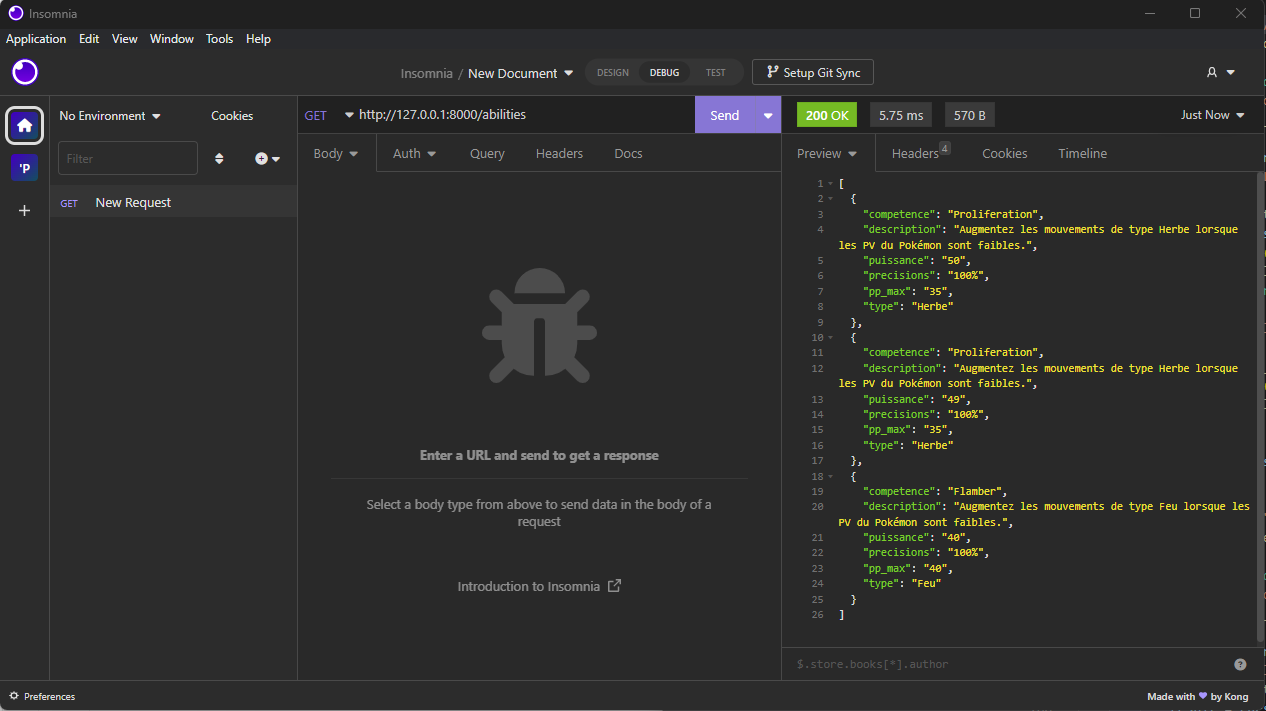


Figure Récupère la liste de toutes les compétences

## <http://127.0.0.1:8000/abilities/1> **Récupère les détails de la compétence précisée par :id**

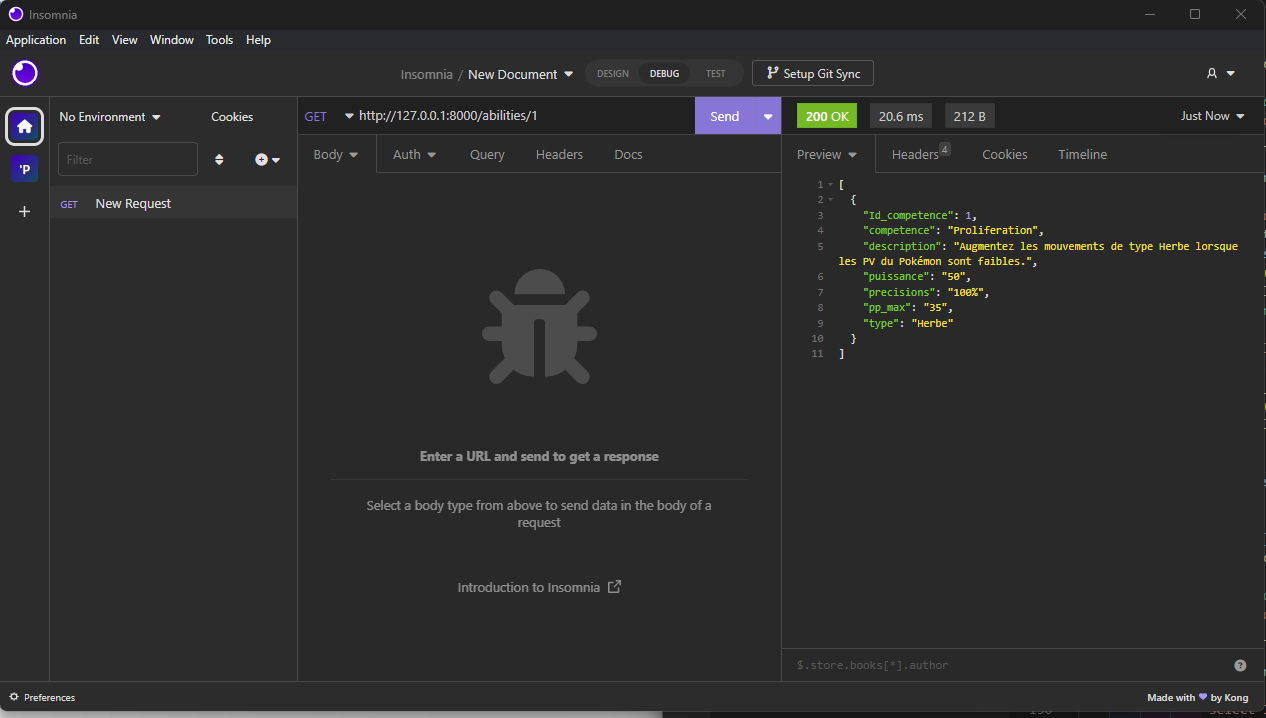


Figure Récupère les détails de la compétence précisée par :id

## <http://127.0.0.1:8000/pokemons/id/types/2> **Récupère le ou les types du pokémon précisé par :id**

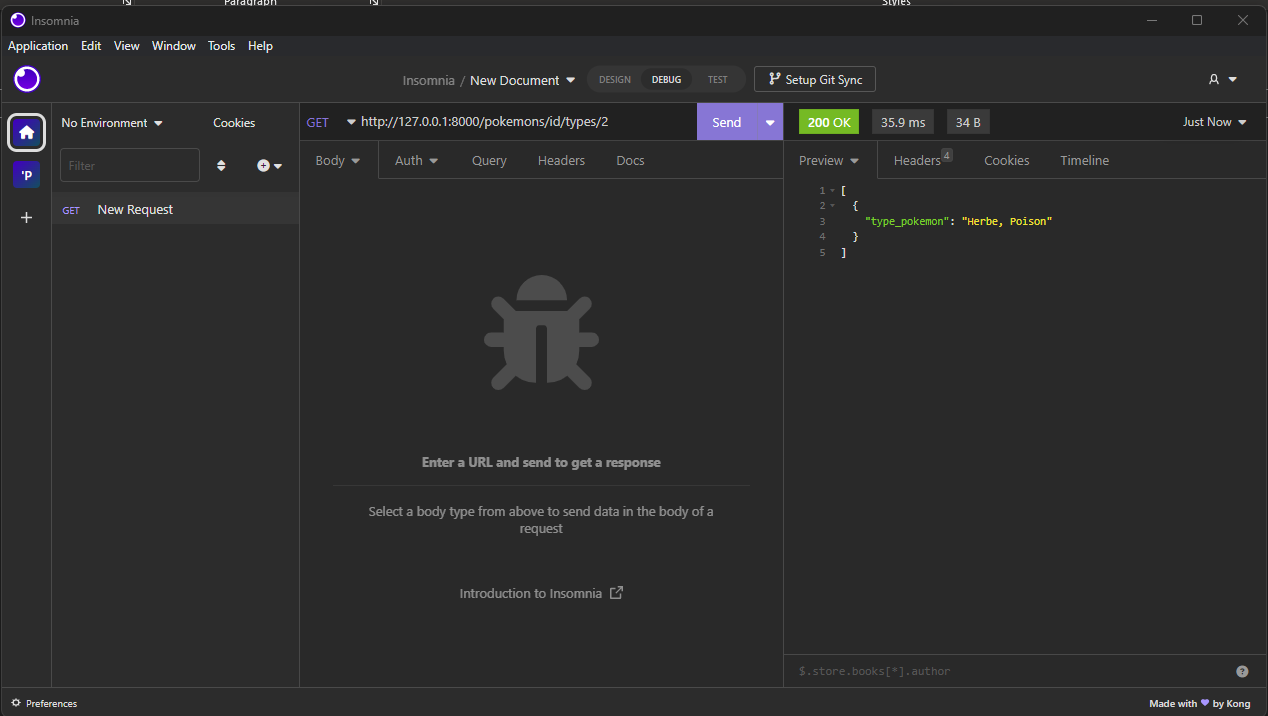


Figure Récupère le ou les types du pokémon précisé par :id

## <http://127.0.0.1:8000/pokemons/id/abilities/2> **Récupère les compétences apprenables du pokémon précisé par :id**

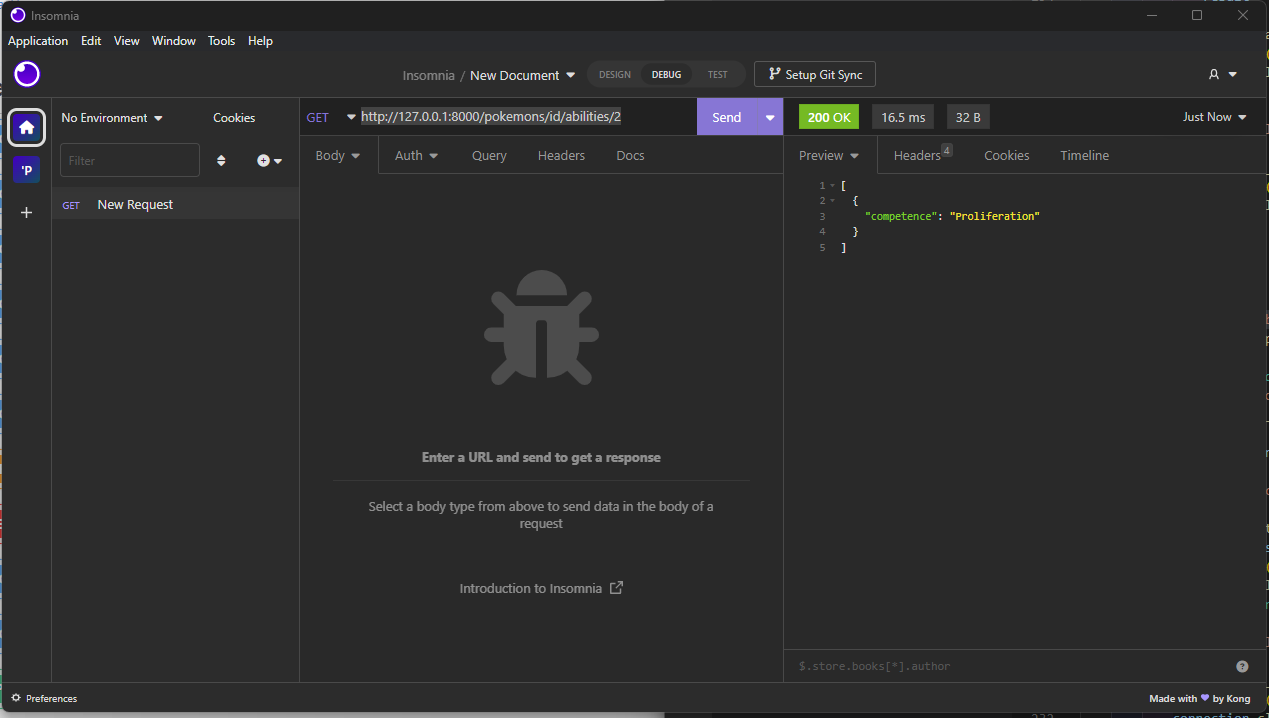


Figure Récupère les compétences apprenables du pokémon précisé par :id

## **Ajout d’un pokémon (À noter pour exécuter les commandes de 9 à 11, il faut les exécuter dans l’ordre 11, 10 et puis 9 à cause des clés étrangères qui relient les 3 tables ensembles)**

### Base de données avant d’ajouter le nouveau pokémon :

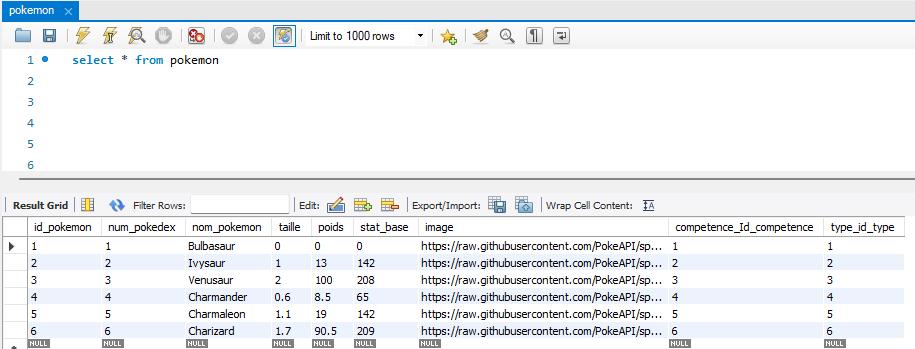


Figure 9. Base de données Ajout d’un pokémon

### On ajoute un nouveau pokémon en utilisant le lien <http://127.0.0.1:8000/pokemons/>

Dans la partie JSON on rajoute ces informations pour le nouveau pokémon :

{

"num\_pokedex": 17,

"nom\_pokemon": "Pidgeotto",

"taille": 3.0,

"poids": 66.1,

"state\_base": 4,

"image": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/17.png"

}

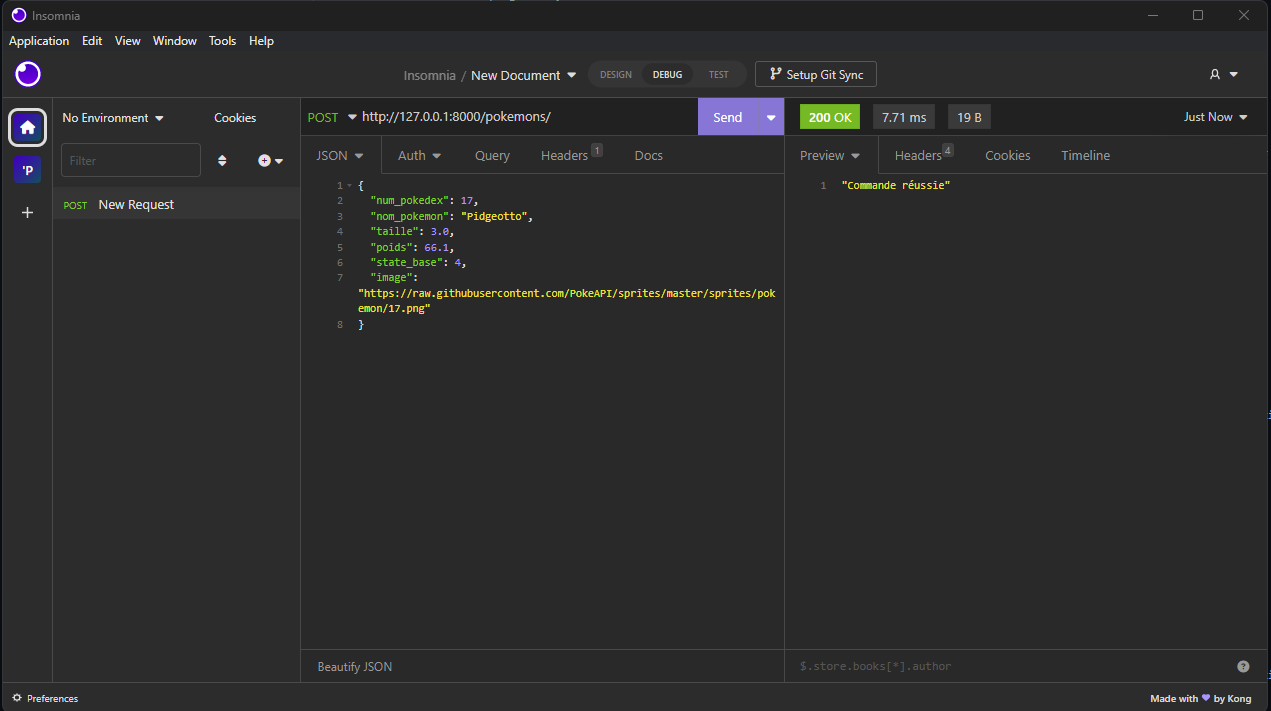


Figure ajoute un nouveau pokémon

### Base de données après d’avoir rajouter le pokémon :

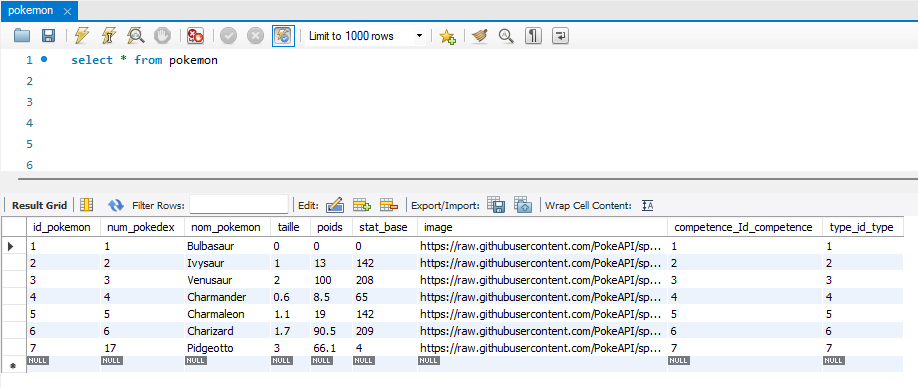


Figure Base de données après d’avoir rajouter le pokémon

## **Ajout d’une compétence**

### Base de données avant de rajouter la nouvelle compétence du pokémon :

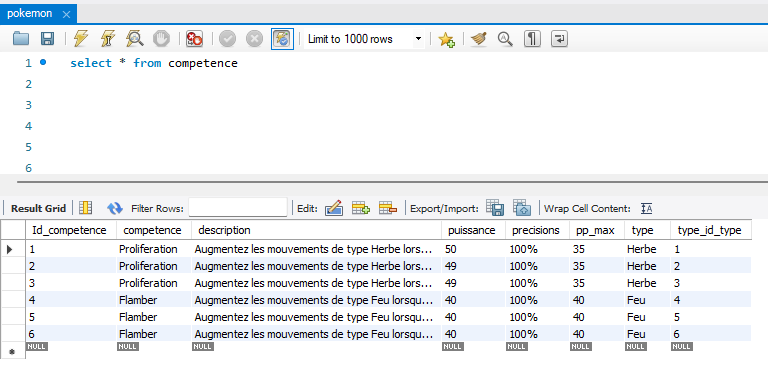


Figure Base de données avant de rajouter la nouvelle compétence du pokémon

### On rajoute une nouvelle compétence en utilisant le lien <http://127.0.0.1:8000/abilities/>

Dans la partie JSON on rajoute ces informations pour la nouvelle compétence :

{

"competence": "Ail D’acier",

"description": "Transforme les ails en acier",

"puissance": "45",

"precision": "75%",

"pp\_max": "30",

"type": "Air, Acier"

}

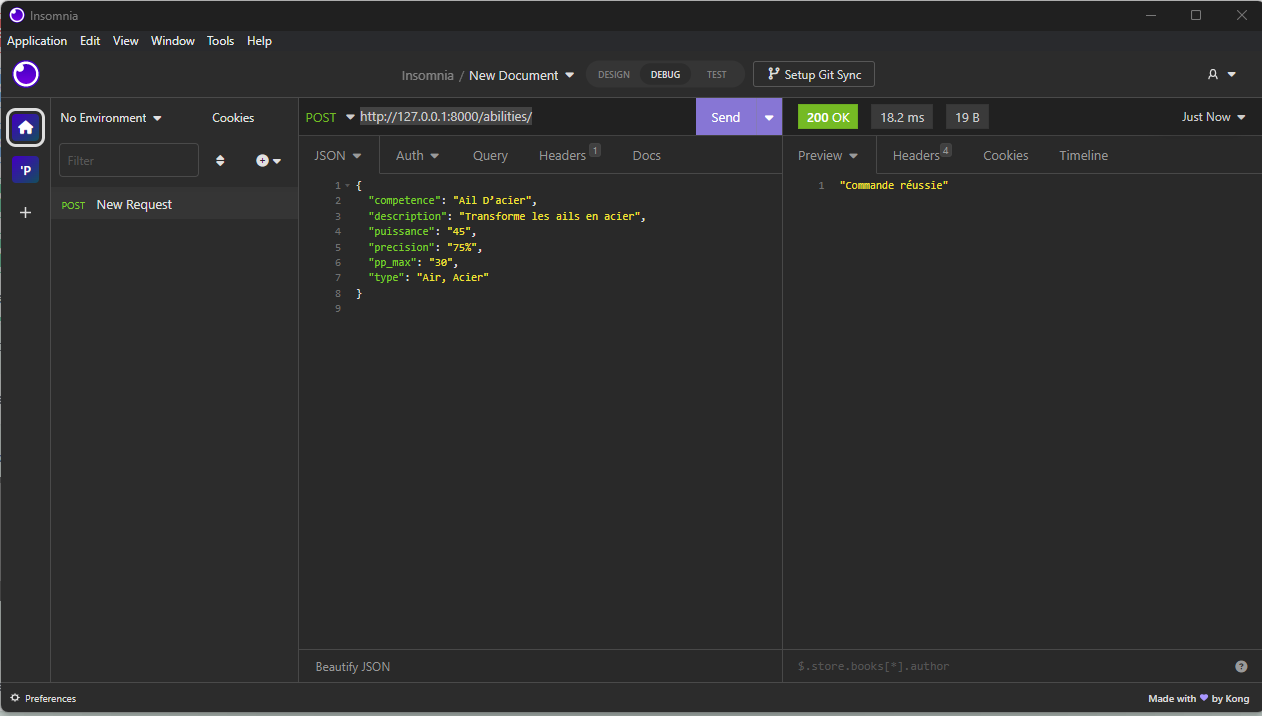


Figure On rajoute une nouvelle compétence

### Base de données après avoir rajouter la nouvelle compétence :

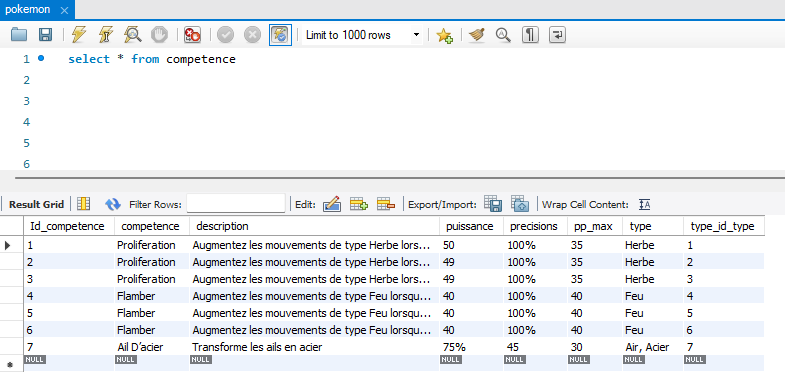


Figure Base de données après avoir rajouter la nouvelle compétence

## **Ajout d’un type**

### Base de données avant de rajouter le nouveau type.

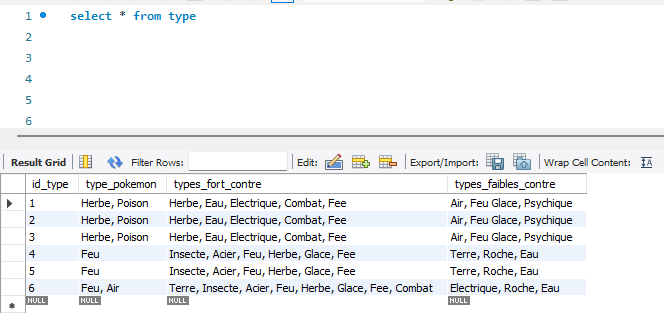


Figure Base de données avant de rajouter le nouveau type.

### On rajoute un nouveau type avec notre API. On utilise le lien <http://127.0.0.1:8000/type/>

Puis dans la partie JSON on ajoute les données :

{

"type\_pokemon": "Acier",

"type\_fort\_contre": "Roche",

"type\_faibres\_contre": "Vole"

}

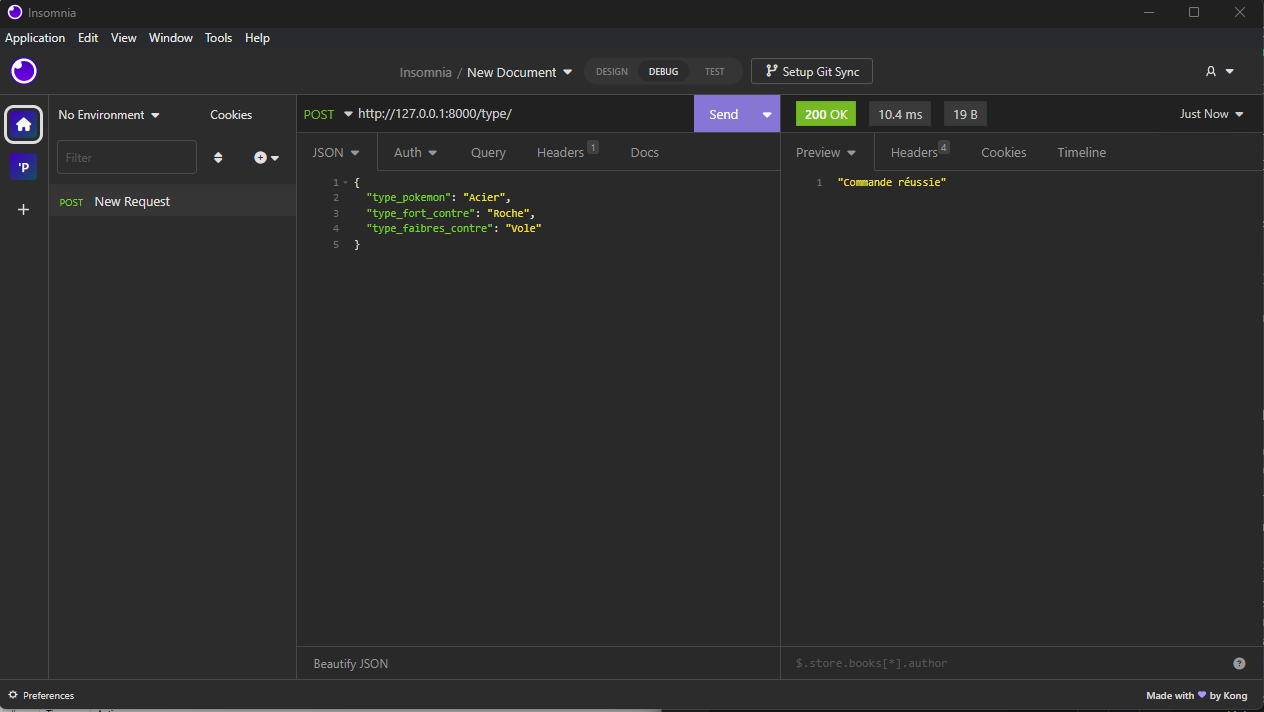


Figure On rajoute un nouveau type avec notre API

### Base de données après avoir rajouter le nouveau type :

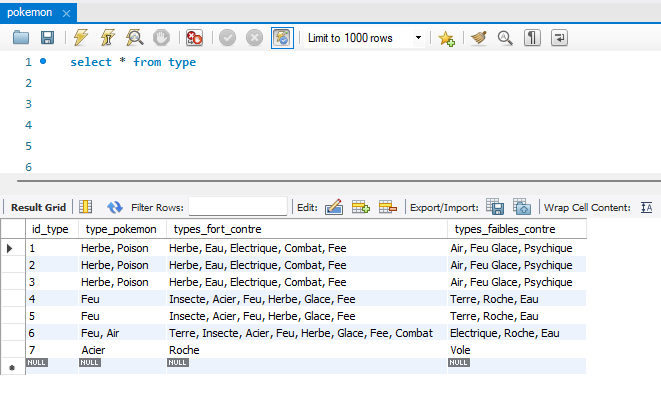


Figure Base de données après avoir rajouter le nouveau type

## **Modification du pokémon précisé par :id**

### Base de données avant modification :

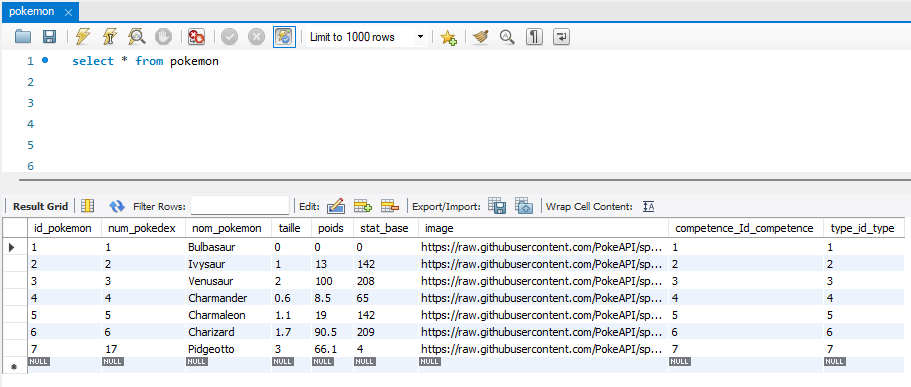


Figure Base de données avant modification

### Pour modifier les informations du pokémon on utilise le lien : <http://127.0.0.1:8000/pokemons/1>

Dans la partie JSON on rajoute :

{

"num\_pokedex": 0,

"nom\_pokemon": "Bulbasaur",

"taille": 34,

"poids": 44,

"state\_base": 555,

"image": "string"

}

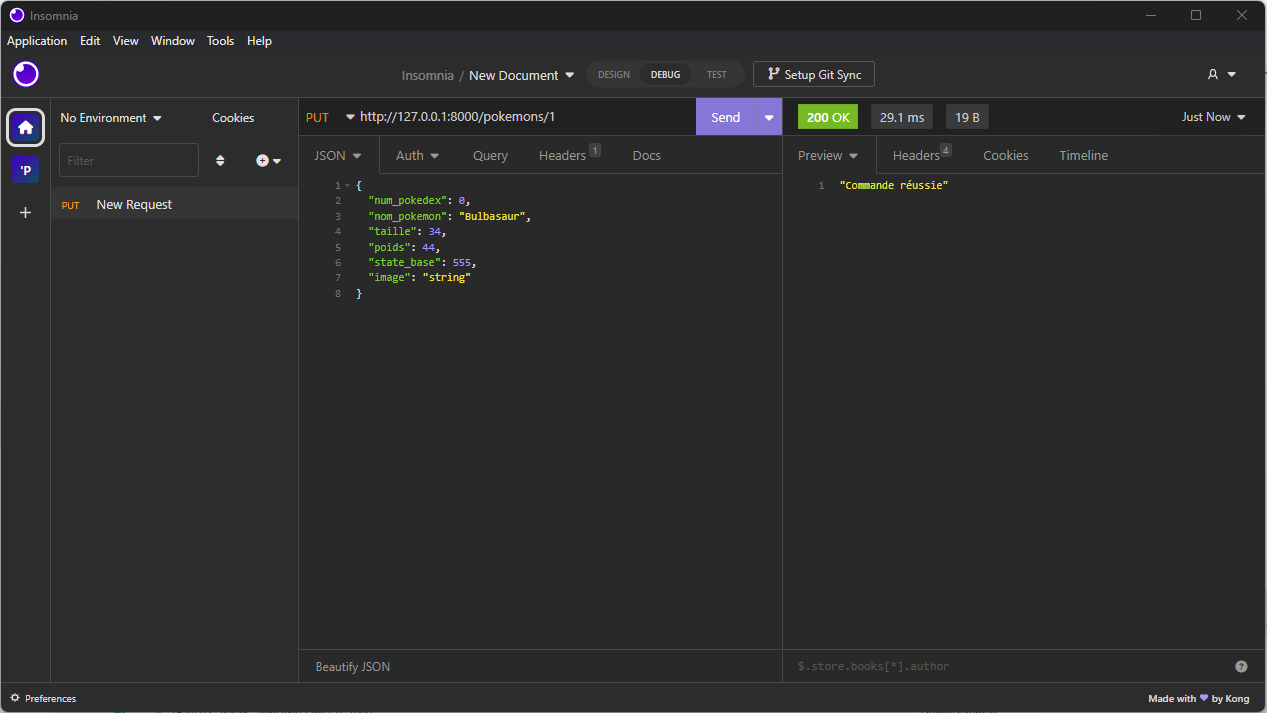


Figure Pour modifier les informations du pokémon

### Base de données après modification :

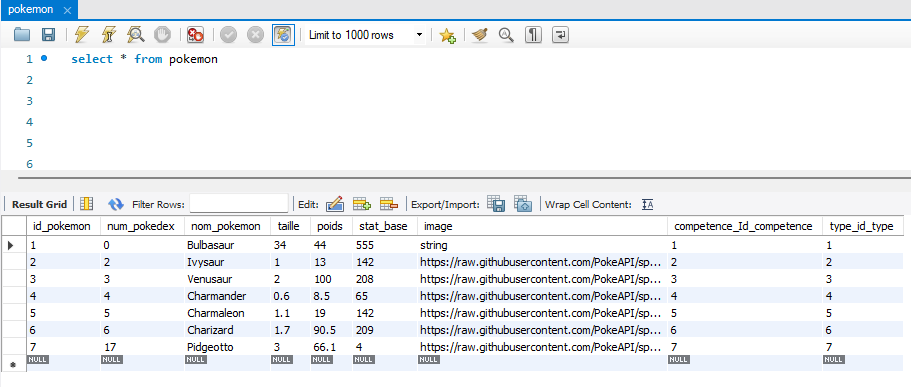


Figure Base de données après modification

## **Modification de la compétence précisée par :id**

### Base de données avant modification :

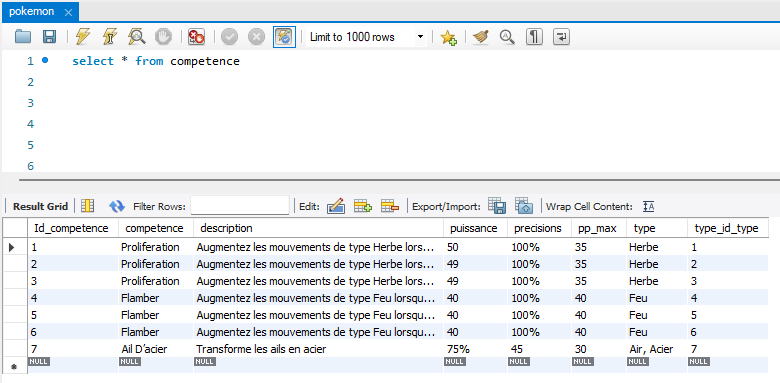


Figure Base de données avant modification

### Pour modifier la compétence on utilise le lien <http://127.0.0.1:8000/abilities/1>

Dans la partie JSON on rajoute :

{

"competence": "string",

"description": "string",

"puissance": "string",

"precision": "string",

"pp\_max": "string",

"type": "string"

}

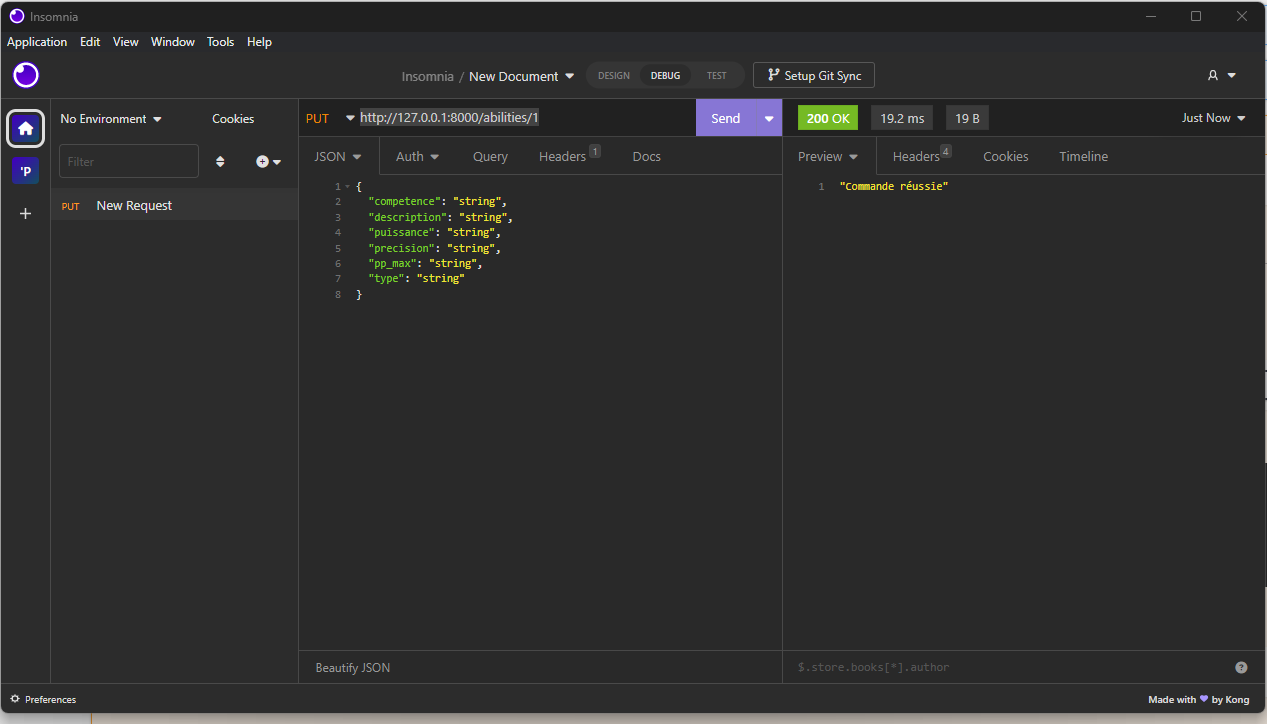


Figure Pour modifier la compétence

### Base de données après modification :

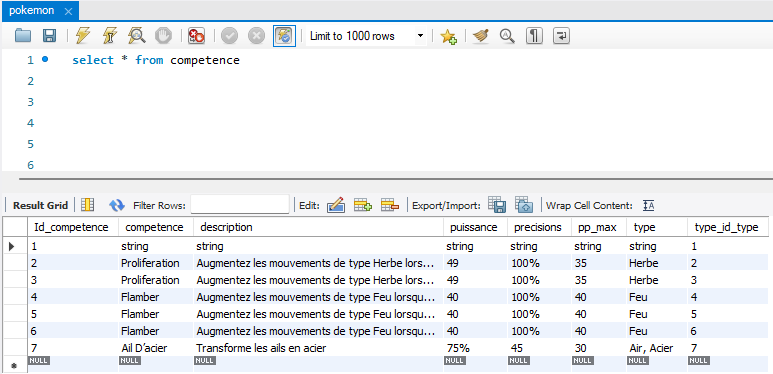


Figure Pour modifier la compétence on utilise le lien

## **Modification du type précisé par :id**

### Base de données avant modification :

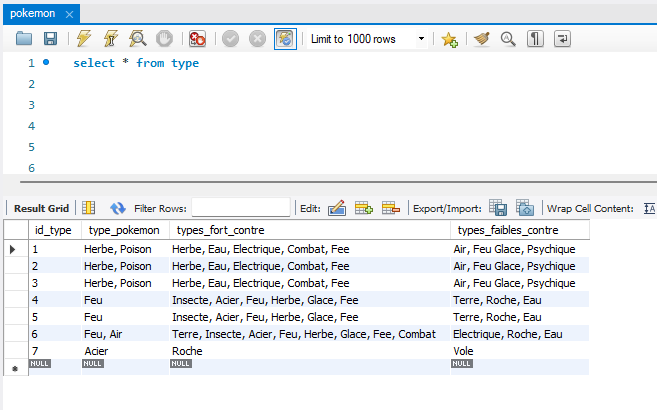


Figure Base de données avant modification

### Pour modifier le type on utilise le lien <http://127.0.0.1:8000/type/1>

Dans la partie JSON on rajoute les informations :

{

"type\_pokemon": "string",

"type\_fort\_contre": "string",

"type\_faibres\_contre": "string"

}

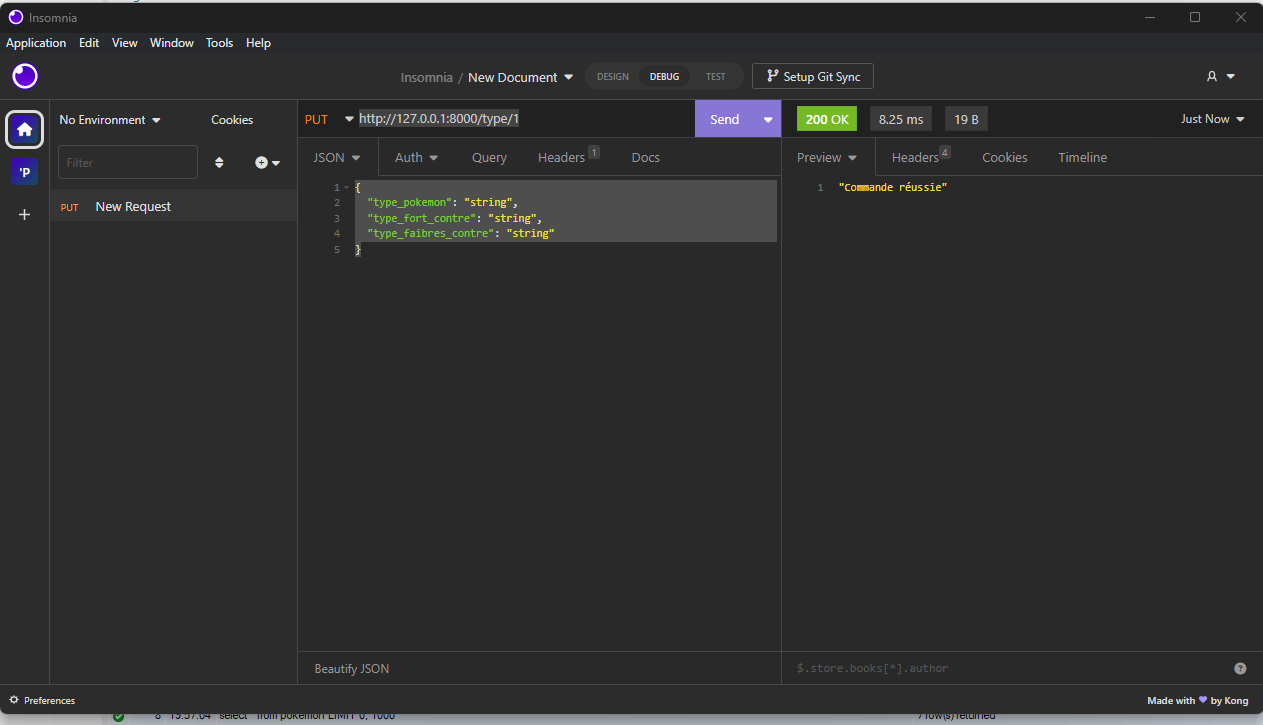


Figure Pour modifier le type on utilise le lien

### La base de données après modification du type :

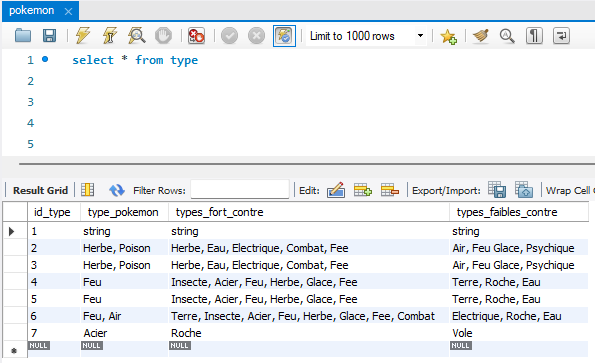


Figure La base de données après modification du type

## **Suppression du pokémon précisé par :id**

### Base de données avant suppression :

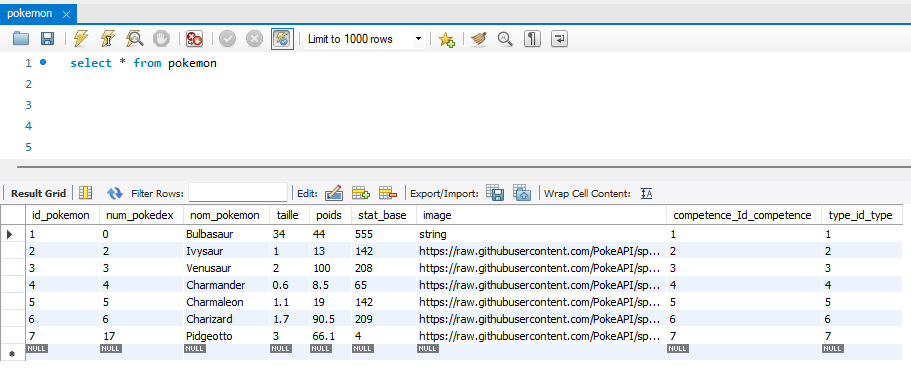


Figure Base de données avant suppression

### On supprime un pokémon en utilisant ce lien <http://127.0.0.1:8000/pokemons/7>

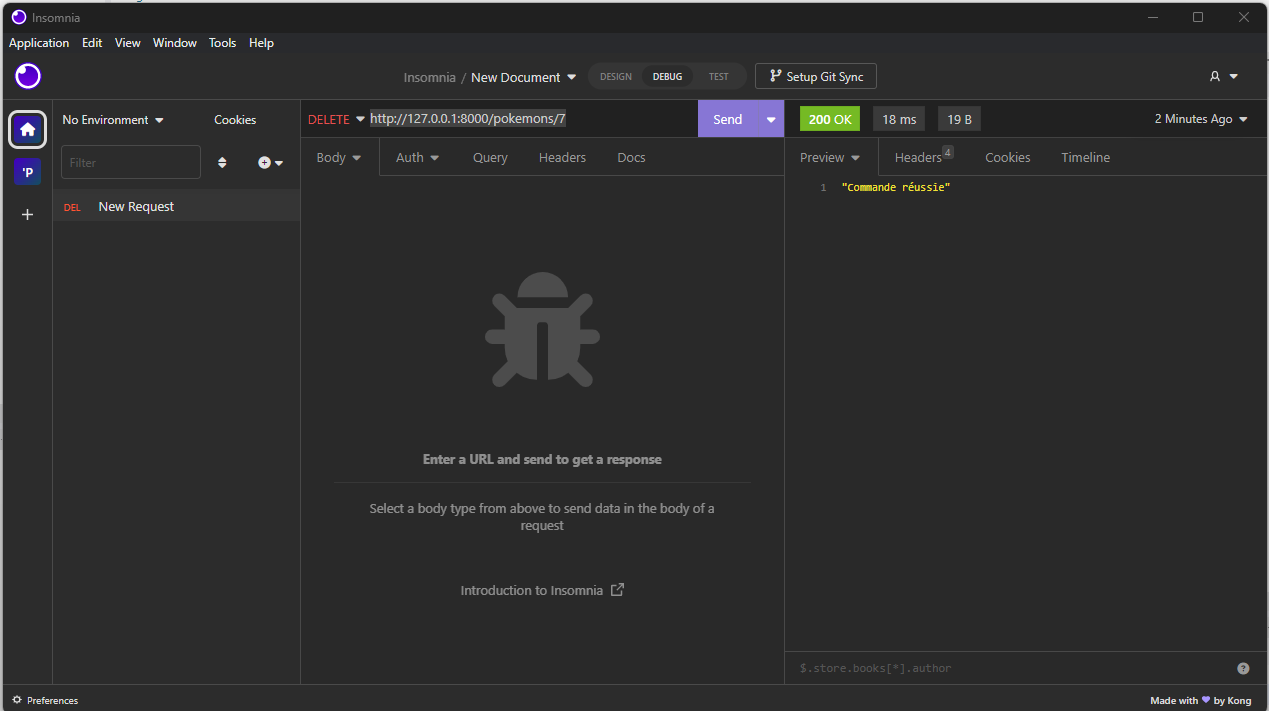


Figure On supprime un pokémon en utilisant

### Base de données après suppression du pokémon :

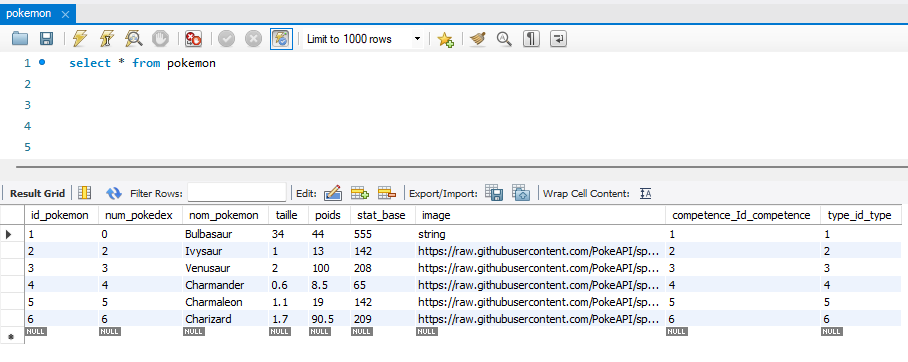


Figure Base de données après suppression du pokémon

## **Suppression de la compétence précisée par :id**

### Base de données avant suppression de la compétence :

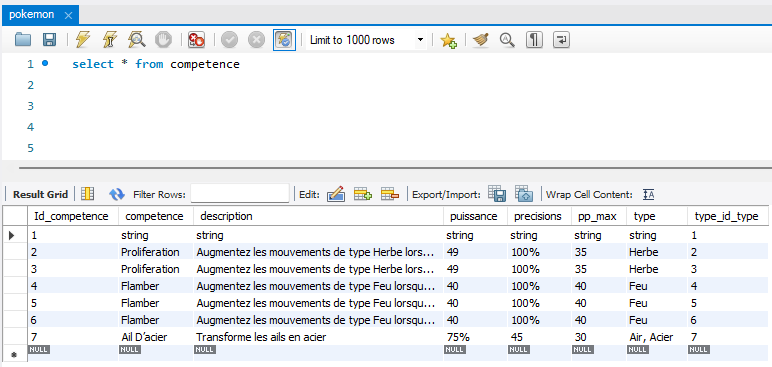


Figure Base de données avant suppression de la compétence

### On supprime la compétence en utilisant le lien <http://127.0.0.1:8000/abilities/7>

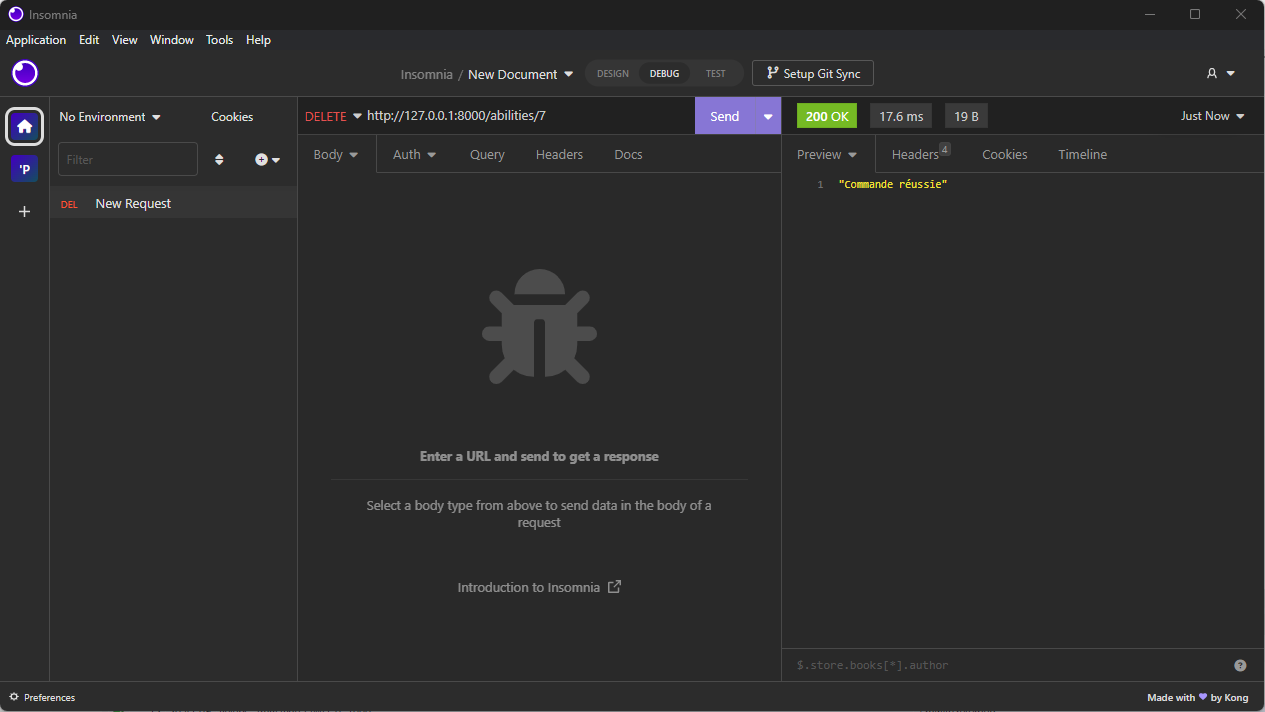


Figure On supprime la compétence

### Base de données après qu’on a supprimé la compétence :

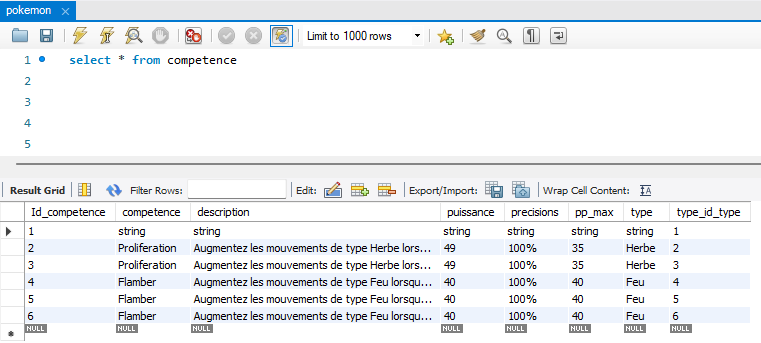


Figure Base de données après qu’on a supprimé la compétence

## **Suppression du type précisé par :id**

### Base de données avant de supprimer le type :

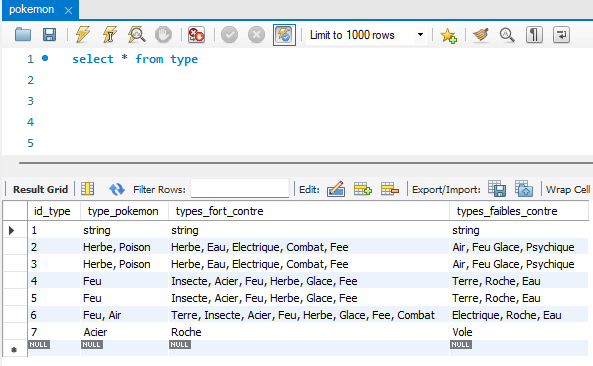


Figure Base de données avant de supprimer le type

### Pour supprimer un type, on utilise le lien : <http://127.0.0.1:8000/type/7>

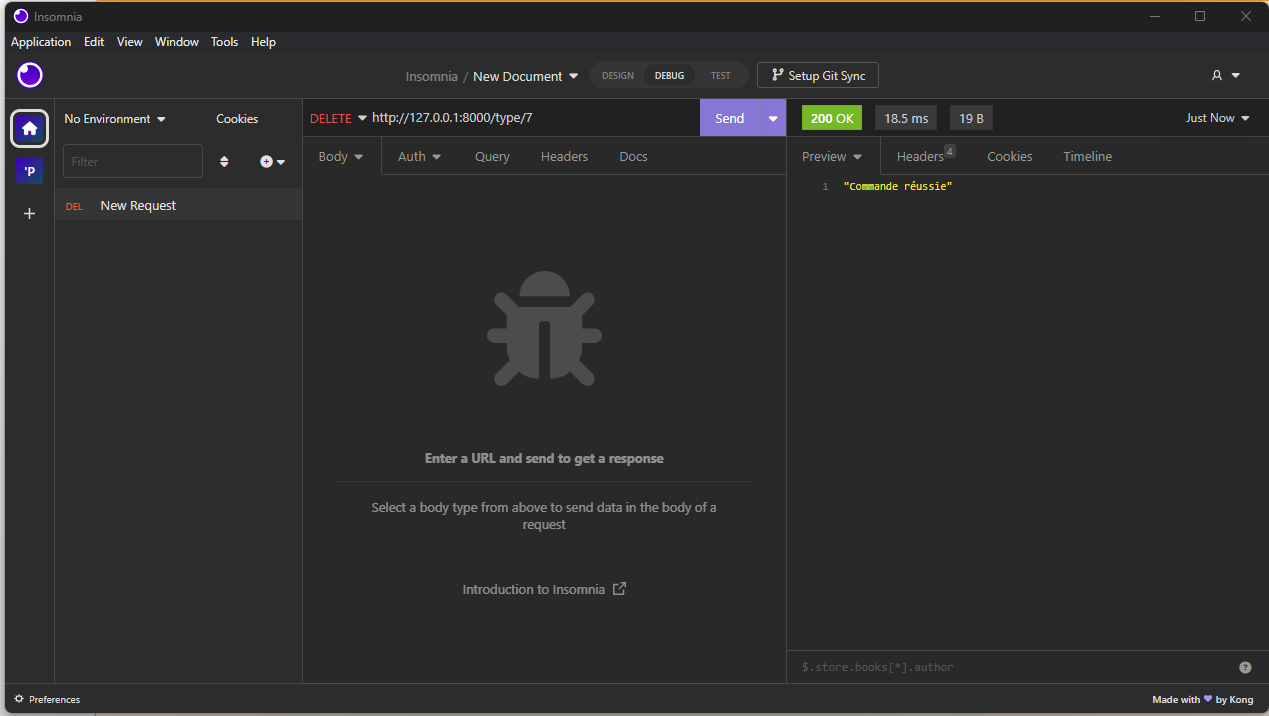


Figure Pour supprimer un type

### Base de données après avoir supprimer le type :

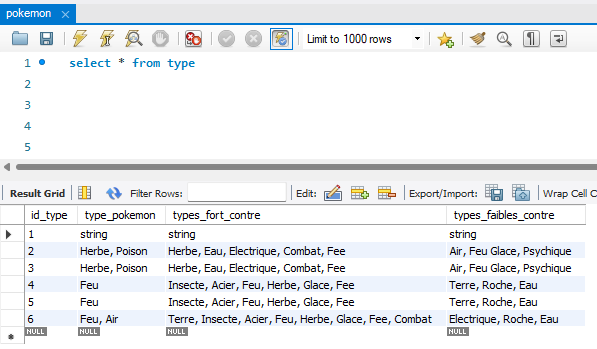


Figure Base de données après avoir supprimer le type

# Conclusion

# Bibliographie

Introduction :

<https://www.astera.com/fr/type/blog/rest-api-definition/#What-is-REST-API>?

Technologies utilisées :

<https://www.mysql.com/products/workbench/#:~:text=MySQL%20Workbench%20is%20a%20unified,Linux%20and%20Mac%20OS%20X.c>

<https://www.lemagit.fr/conseil/MySQL-Community-Server-VS-Oracle-MySQL-quelles-sont-les-differences#:~:text=MySQL%20Community%20Server%20(ou%20Community,triggers%20et%20les%20proc%C3%A9dures%20stock%C3%A9es>.

https://pulsanova.com/blog/tuto-insomnia

MySQL WorkBench:

<https://dev.mysql.com/downloads/workbench/>

MySQL Community Server:

<https://dev.mysql.com/downloads/mysql/>

Insomnia :

<https://insomnia.rest/download>

Python :

<https://www.python.org/downloads/>