EASY WAY TO LEARN

LEARN AND CREATE YOUR FIRST API USING PYTHON AND FLASK

BY ISMAIL CISSE

COPYRIGHT © 2024 ISMAIL CISSE
ALL RIGHTS RESERVED.

TOOLS TO DOWNLOAD BEFORE STARTING:

PYTHON

POSTMAN

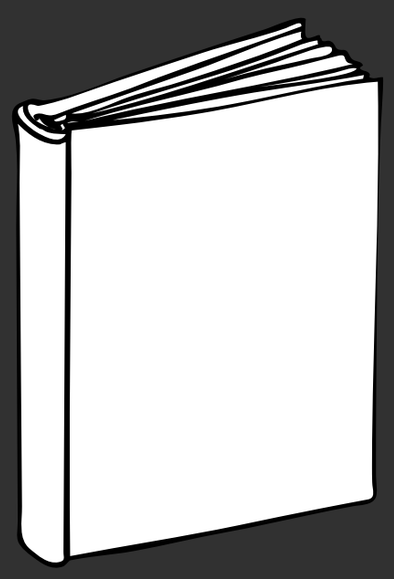
VISUAL STUDIO CODE

WHAT IS AN API?

DO YOU LIKE READING? WELL, WHETHER YOU DO OR DON'T, IT'S OKAY—BOOKS WON'T JUDGE YOU,

😄 BUT THEY MIGHT JUST CHANGE YOUR MIND!

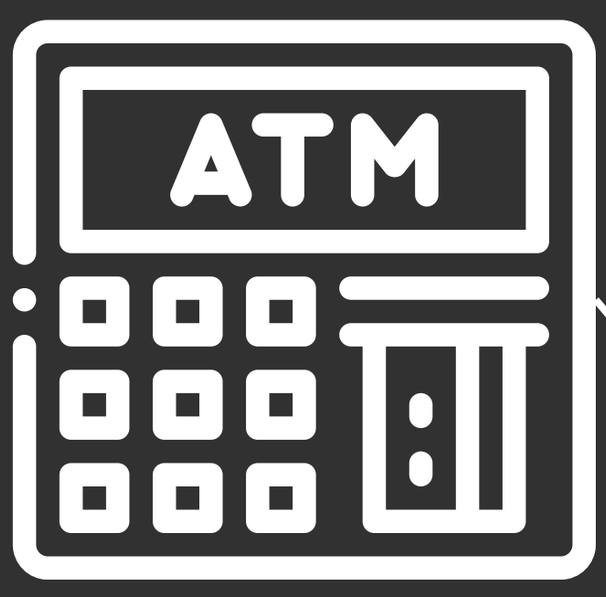
AN API IS LIKE THE LIBRARIAN AT A LIBRARY IT TAKES YOUR REQUESTS(WHAT BOOK YOU WANT)

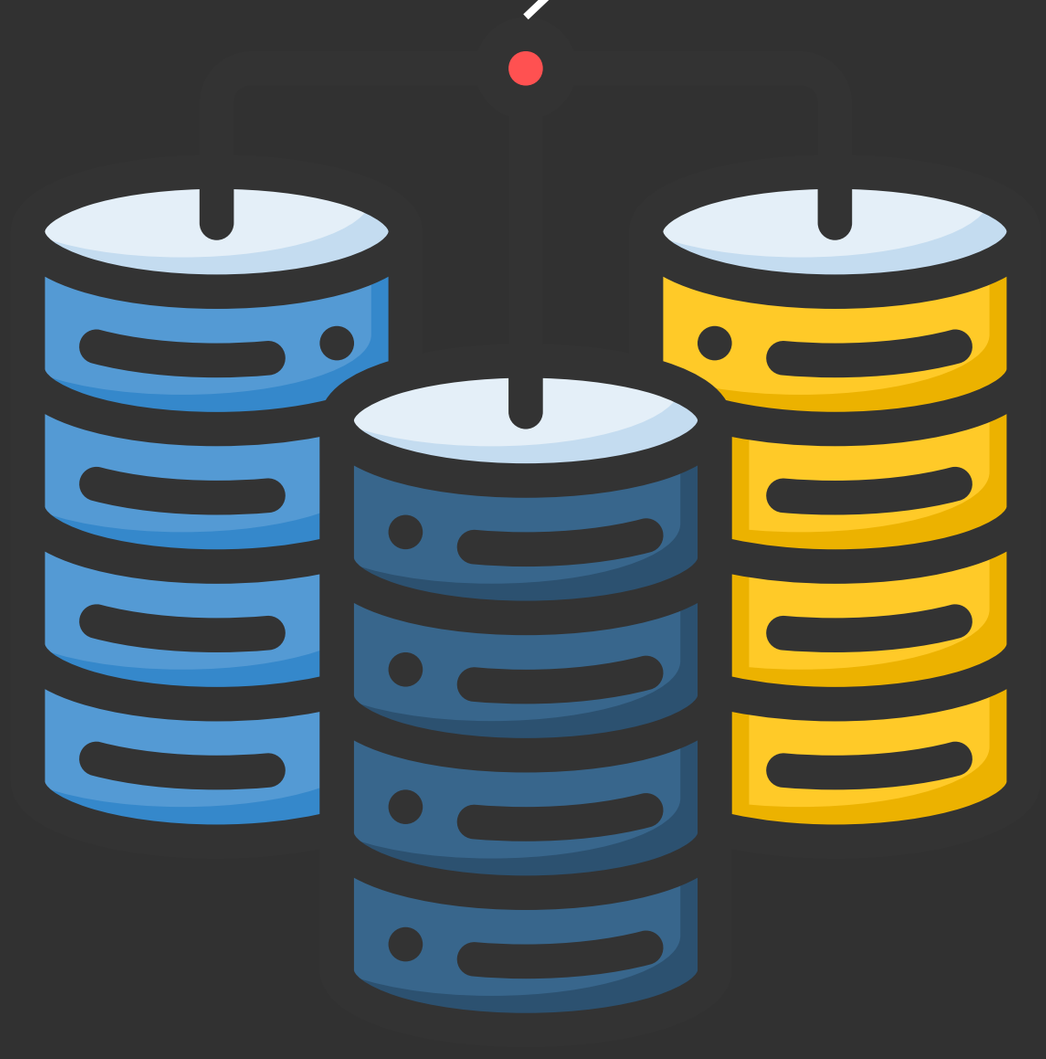


COMMUNICATE WITH THE STACKS (SERVER/DATABASE)

AND BRINGS BACK THE RESPONSE(YOUR BOOK) ALL WHILE YOU COMFORTABLY WAIT A THE COUNTER (APPLICATION).

A REAL LIFE EXAMPLE IS LIKE USING AN ATM.





BANK SERVER

- YOU AS THE USER INTERACT WITH THE ATM'S INTERFACE
- THE ATM ACTS AS A CLIENT, IT SENDS REQUEST TO THE BANK'S API.
- THE BANK'S API WILL PROCESS THE REQUEST, COMMUNICATES WITH THE BANK'S DATABASE, AND SENDS BACK A RESPONSE TO THE ATM.

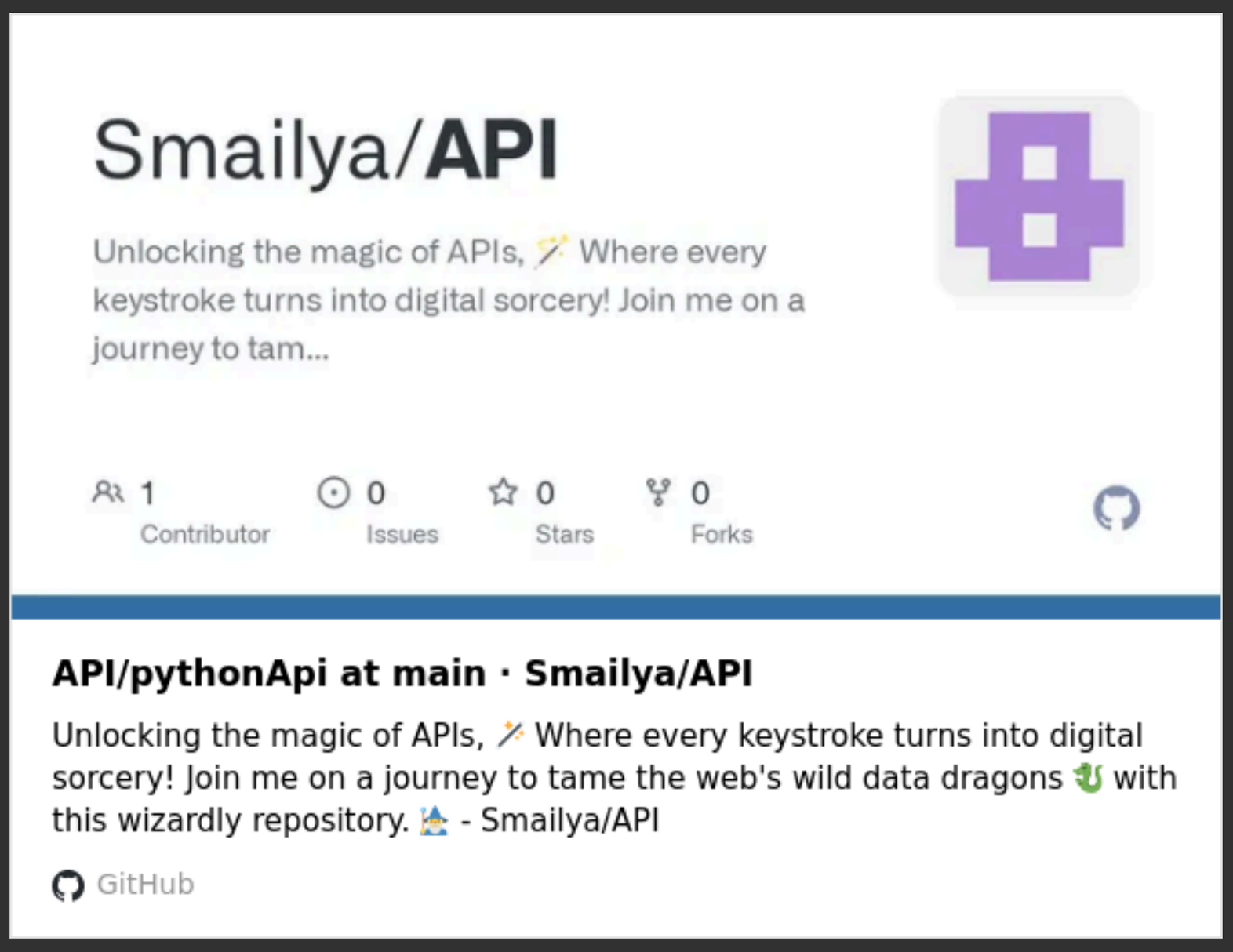
HTTP(HYPER, TEXT TRANSFER PROTOCOL) IS USED TO TRANSMIT DATA OVER THE INTERNET AND DEFINES HOW WEB SERVERS AND BROWSERS SHOULD RESPOND TO VARIOUS COMMANDS. THE MOST POPULAR COMMANDS ARE:

GET #REQUEST DATA FROM A RESOURCE
POST #CREATE A RESOURCE
PUT #UPDATE A RESOURCE
DELETE #DELETE A RESOURCE

GET READY TO EMBARK ON THIS EPIC JOURNEY I'LL BE GANDALF THE GREAT, GUIDING OUR WAY TO THE MISTY CODING, AND YOU'RE OUR BRAVE BILBO BAGGINS!

DOUBLE CLICK THE IMAGE OR GO TO THE FOLLOWING LINK TO DOWNLOAD THE ENTIRE CODE FROM MY GITHUB.

HTTPS://GITHUB.COM/SMAILYA/API/TREE/MAIN/PYTHONAPI



DOWNLOAD THE FILE main.py

AFTER DOWNLOADING YOU WILL SEE THAT I EXPLAINED EACH LINE OF CODE TO MAKE IT EASY FOR YOU TO UNDERSTAND WHAT'S REALLY GOING ON.

NOW OPEN VISUAL STUDIO

CREATE A NEW FOLDER YOU NAME IT API

CREATE A NEW FILE NAME IT main.py

OPEN YOUR TERMINAL AND TYPE:

```
pip install flask
```
ENTER

IT WILL INSTALL THE FLASK FRAMEWORK FOR YOU, FLASK IS THE SERVER FROM WHICH YOU WILL BE RUNNING YOUR API

NOW I WANT YOU TO TYPE OUT EVERYTHING FROM THE FILE main.py

RESIST THE URGE TO COPY-PASTE! LET THOSE FINGERS DANCE ON THE KEYS—TYPING IS THE PATH TO TRUE UNDERSTANDING!

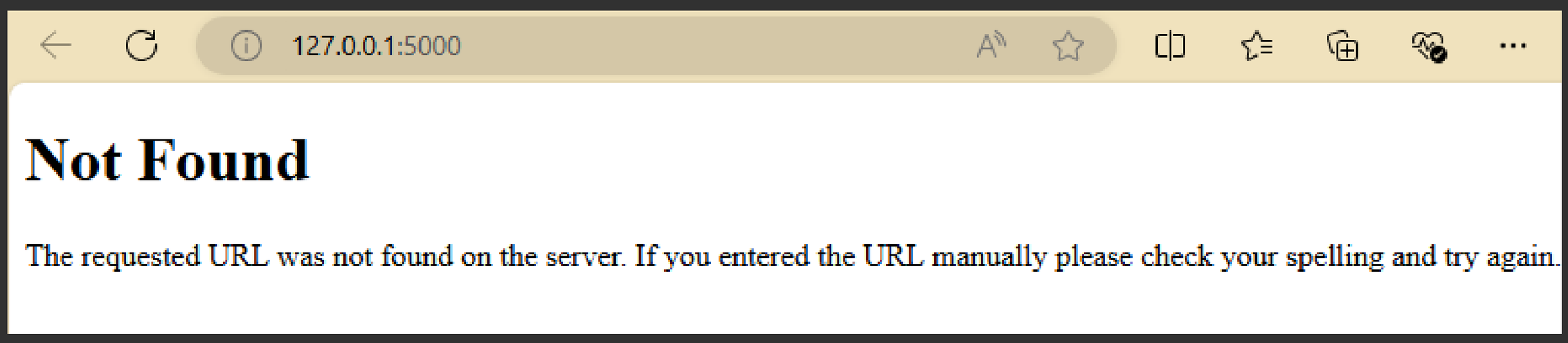
WHEN YOU DONE TYPING TO RUN THE SERVER TYPE

```
\API> python main.py
```
ENTER

YOU WILL SEE A URL SIMILAR TO THE ONE BELOW OPEN IT IN YOUR BROWSER

```
* Running on http://127.0.0.1:5000
```

OPEN YOUR BROWSER YOU WILL SEE THE FOLLOWING:



IT'S BECAUSE YOU DON'T HAVE A HOME ROUTE IF YOU WANT YOU CAN DEFINE A FUCTION FOR THAT. BY TYPING

```
@app.route("/", methods=["GET"])
def home():
    return "home"
```

EXAPLANATIONS OF THE CODE:

@app.route USING THE ROUTE() DECORATOR TO BIND A FUNCTION TO A URL.

("/", METHODS=["GET"]) INDICATES THAT THE FOLLOWING FUNCTION HANDLES GET REQUESTS TO THE ROOT URL ("/")

def home(): # DEFINING A VIEW FUNCTION NAMED 'HOME' THAT WILL BE CALLED WHEN THE ROOT URL IS ACCESSED.

return "home" # RETURNING A SIMPLE STRING "HOME" AS THE RESPONSE TO THE GET REQUEST.

RUN THE SERVER AGAIN YOU WILL SEE IT DISPLAY HOME. NOW ADD THE FOLLOWING TO YOUR URL: get-user/422

IT DOESN'T MATTER WHAT USER_ID IS PUT IN THE URL PATH BECAUSE THE GET_USER FUNCTION IS HARD-CODED TO RETURN THE SAME DATA FOR ANY USER_ID. SPECIFICALLY, THE USER_DATA DICTIONARY ALWAYS CONTAINS THE SAME NAME AND EMAIL VALUES REGARDLESS OF THE USER_ID. ONLY THE USER_ID FIELD IN THE RESPONSE CHANGES TO REFLECT THE USER_ID PROVIDED IN THE URL.

YOU WILL SEE THE FOLLOWING JSON RESPONSE IN YOUR BROWSER OR API CLIENT:


```
{
  "email": "Bilbo.baggains@example.com",
  "name": "Bilbo Baggins",
  "user_id": "2"
}
```


THE FLASK APPLICATION MATCHES THE URL /GET-USER/2 TO THE ROUTE DEFINED BY THE @APP.ROUTE("/GET-USER/<USER_ID>") DECORATOR.

THE GET_USER FUNCTION IS CALLED WITH USER_ID SET TO 2.

INSIDE THE GET_USER FUNCTION, A DICTIONARY USER_DATA IS CREATED WITH THE KEYS "USER_ID", "NAME", AND "EMAIL", AND THEIR CORRESPONDING VALUES.

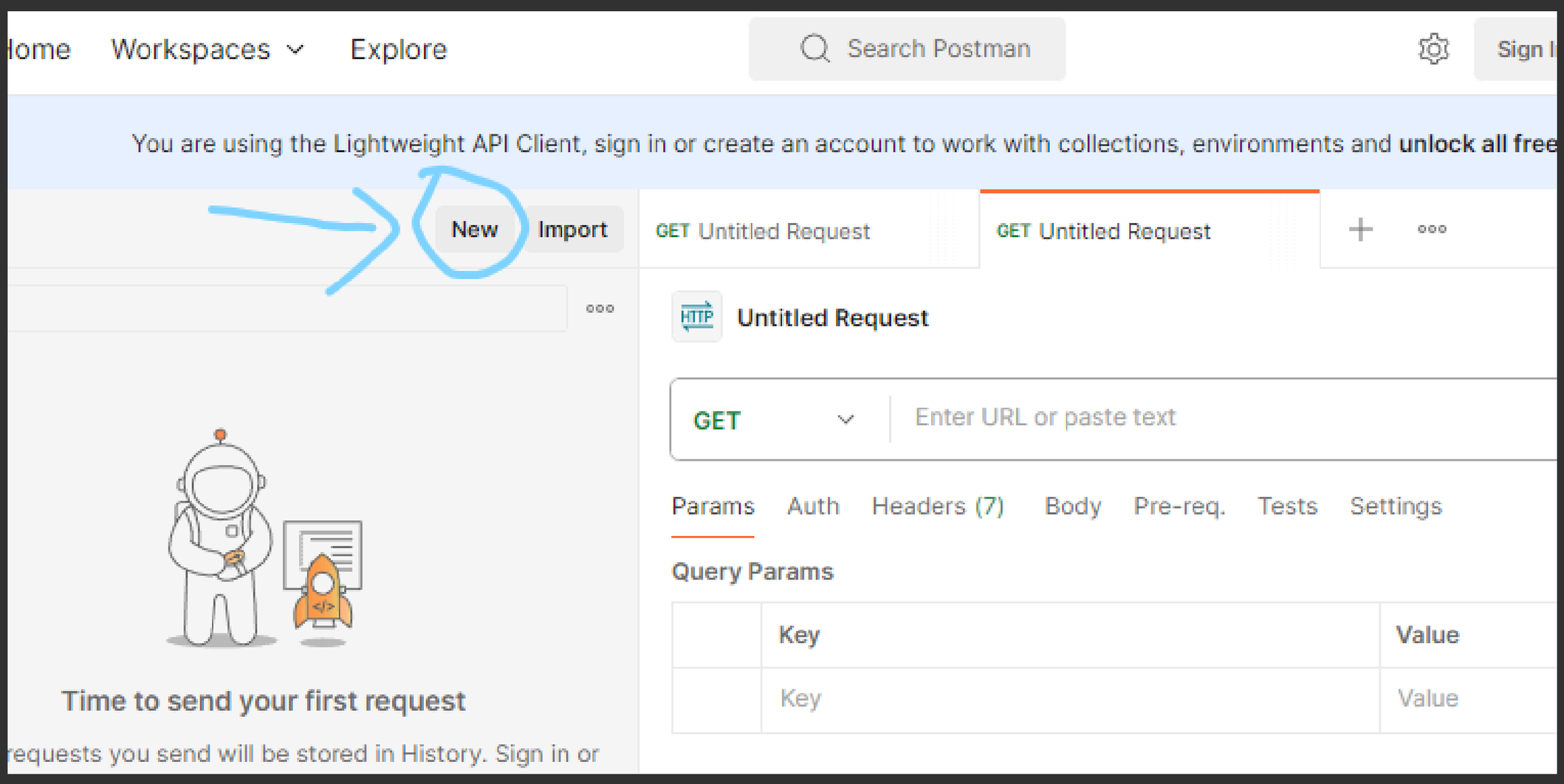
THE FUNCTION CHECKS FOR AN OPTIONAL QUERY PARAMETER EXTRA USING REQUEST.ARGS.GET("EXTRA"). IF THE EXTRA PARAMETER IS PROVIDED IN THE REQUEST, IT IS ADDED TO THE USER_DATA DICTIONARY.

THE USER_DATA DICTIONARY IS CONVERTED TO A JSON RESPONSE USING JSONIFY(USER_DATA) AND RETURNED TO THE CLIENT WITH A STATUS CODE 200 (OK).

IF YOU INCLUDE AN EXTRA QUERY PARAMETER IN THE REQUEST, SUCH AS HTTP://127.0.0.1:5000/GET-USER/2?EXTRA=SOME_VALUE, THE JSON RESPONSE WILL INCLUDE THE EXTRA FIELD:

ALRIGHT, FOLKS, BRACE YOURSELVES—IT'S TIME TO SUMMON OUR MYSTICAL POWERS AND UNLEASH OUR MAGIC WAND API TESTER, POSTMAN! 🧙‍♂️🔮

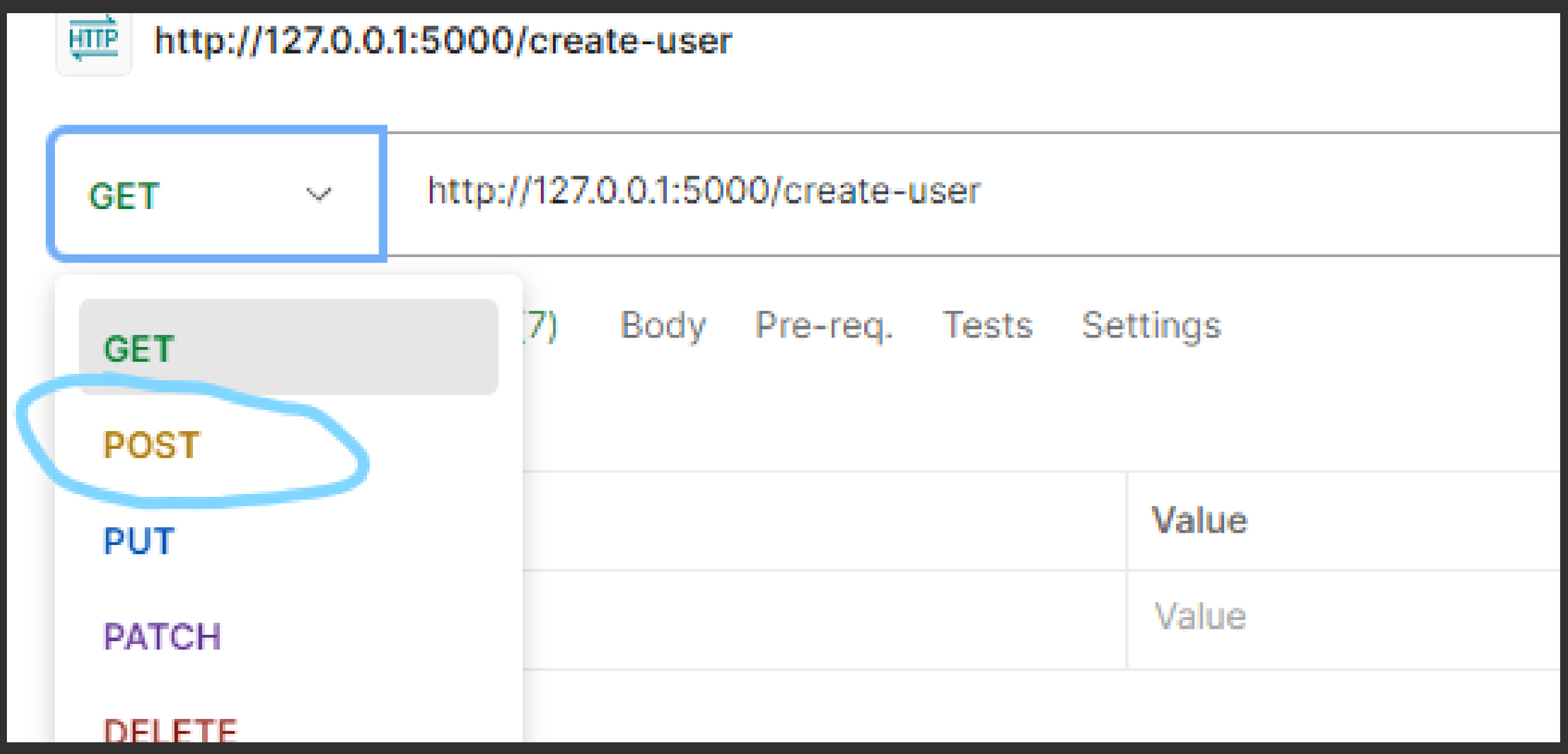
CLICK ON NEW THEN SELECT HTTP



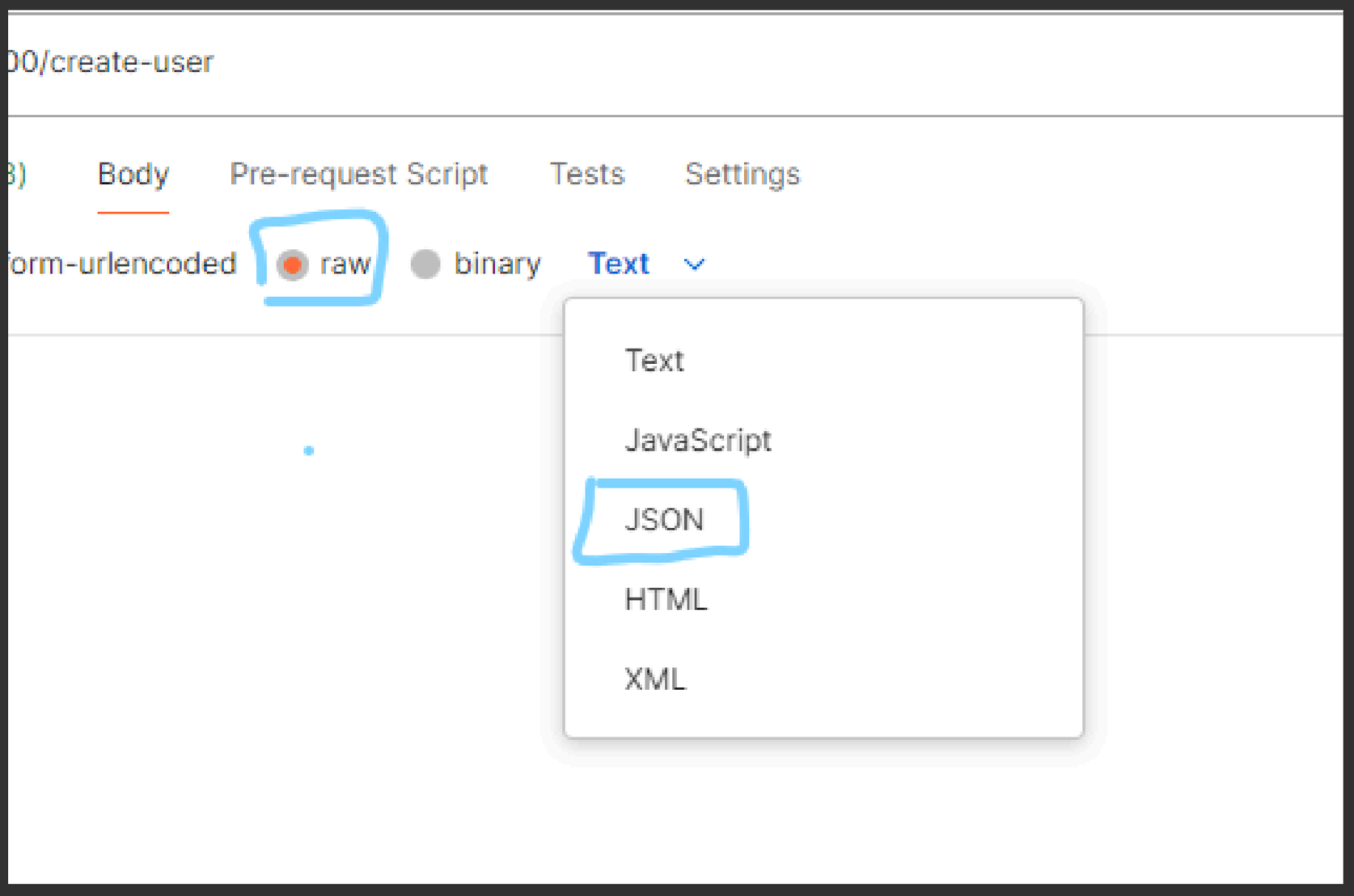
COPY AND PASTE YOUR LOCAL SERVER URL INTO POSTMAN.

ADD THE FOLLOWING TO IT: /CREATE-USER

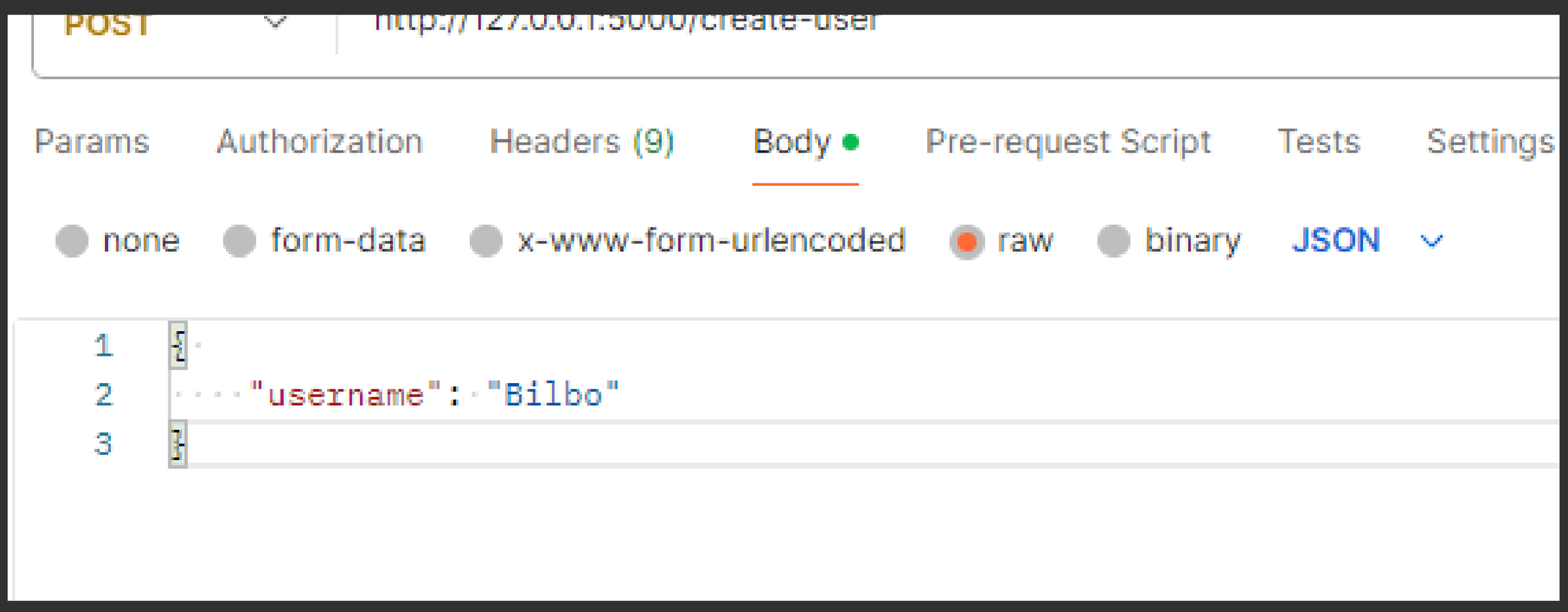
CHANGE THE METHOD TO BE POST



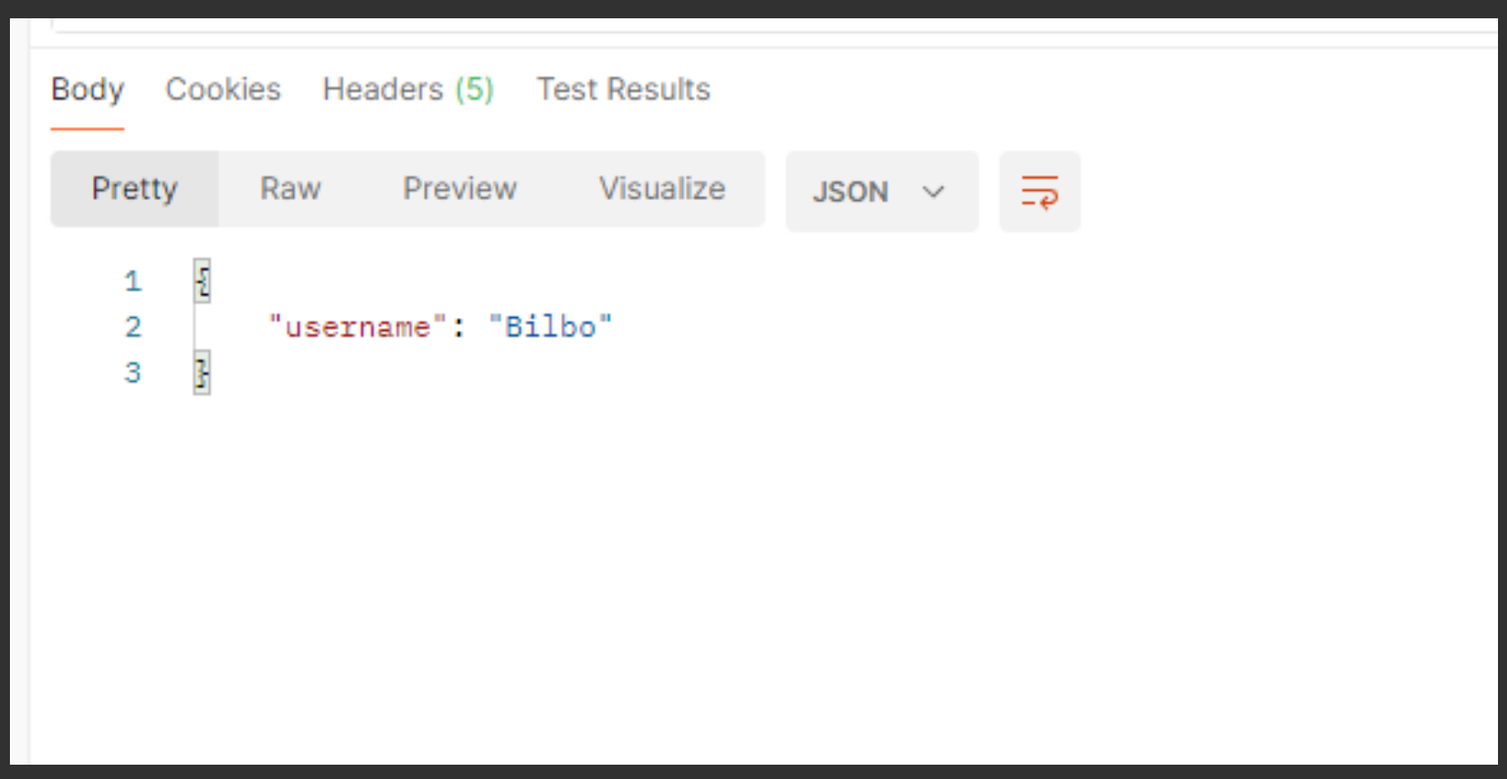
THEN GO TO BODY TO INSERT JSON DATA, SELECT RAW, CHANGE TEXT TO JSON, YOU CAN PASS A JSON OBJECT THE FOLLOWING WAY



YOU CAN PASS A JSON OBJECT THE FOLLOWING WAY



PRESS SEND, YOU WILL SEE SOMETHING DO

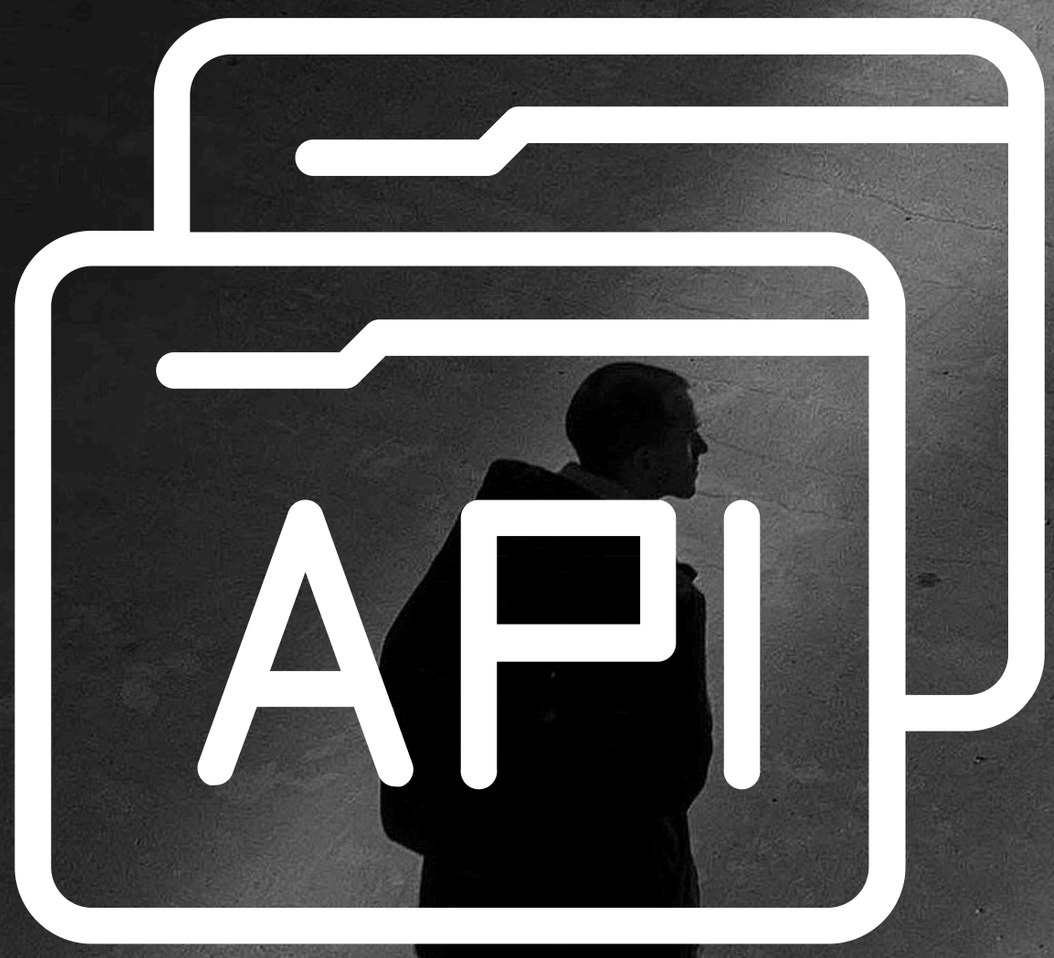


CONGRATULATIONS ON COMPLETING THE BOOK!

AS A FELLOW DEVELOPER, I KNOW THAT CODING CAN SOMETIMES BE CHALLENGING. BUT DON'T LET THAT STOP YOU! KEEP GOING, KEEP PUSHING FORWARD. THE MORE YOU PRACTICE, THE BETTER YOU'LL BECOME AT IT. ONE THING THAT HAS HELPED ME IS TO TRY TO HAVE FUN WHILE CODING. WHEN YOU ENJOY THE PROCESS, IT CAN MAKE THE CHALLENGES FEEL LESS DAUNTING.

I SPENT A LOT OF TIME PUTTING THIS BOOK TOGETHER I HOPE THAT YOU FOUND A LOT OF VALUE FROM IT. FEEL FREE TO SHARE IT.

IF YOU HAVE ANY QUESTIONS ABOUT THE BOOK OR PROGRAMMING IN GENERAL, MY DOOR IS ALWAYS OPEN. YOU CAN REACH ME ON MY LINKEDIN PROFILE AT HTTPS://WWW.LINKEDIN.COM/IN/ISMAIL-CISSE/. I'M HERE TO HELP YOU, SO DON'T HESITATE TO CONNECT WITH ME IF YOU NEED ANY ASSISTANCE.


API