

How ChatGPT Turned Me into an Interactive Console

👤 Michal Kovacik ⌚ March 28, 2023, 7:54 AM



The journey with a software engineer who becomes an interactive console for ChatGPT, an AI language model. Witness code, collaboration, and AI-powered guidance as they tackle GitLab APIs and Python code. Explore the potential and limitations of AI-human collaboration in this journey. Discover how the right guidance can conquer complex problems!

It was a rainy Sunday, a day when most people would choose to stay indoors, warm and cozy. As the leader of a team of software engineers, and a developer at heart, I decided to combine work and play, embarking on a journey to the digital realm of ChatGPT.

The adventure began with a simple query, echoing through the virtual landscape: "Can you recommend some interesting statistics I can create based on data from the GitLab API?" The AI oracle, ChatGPT, responded swiftly, offering a treasure trove of insights. Among the many options, I chose to focus on one specific statistic: "Projects with the longest open issues or merge requests."

MK

can you recommend some interesting statics which i can create base on data from gitlab API <https://docs.gitlab.com/ee/api/rest/index.html>

Sure! GitLab's API provides a wealth of information about projects, users, and various other aspects of the platform. Here are some interesting statistics you can create using the GitLab API:

1. Project Activity:

- Most active projects (based on commit frequency, merge requests, or issues)
- Most starred/forked projects
- Projects with the longest open issues or merge requests
- Distribution of project creation over time (monthly or yearly)
- Distribution of programming languages used in projects

2. User Activity:

- Most active users (based on commits, issues, or merge request activity)
- Users with the highest number of contributions to other projects
- Users with the highest number of followers
- Distribution of user registration over time (monthly or yearly)
- Geographic distribution of users (if location data is available)

3. Merge Requests & Issues:

- Average time taken to close a merge request or issue
- Most frequently used labels for merge requests or issues
- Distribution of merge request or issue status (open, closed, merged) across projects
- Correlation between project activity and the number of open issues or merge requests

4. Repository Statistics:

- Distribution of repository sizes (number of files, lines of code, or bytes)
- Most common file types in repositories
- Most frequently updated files in a project (based on commit history)

5. Continuous Integration (CI) & Continuous Deployment (CD):

- Average build time for projects using GitLab CI/CD
- Most common build statuses (success, failed, canceled, etc.)
- Distribution of build environments and tools used in projects

Remember to respect GitLab's API rate limits and user privacy when collecting and analyzing data.

🔄 Regenerate response

As I dove deeper into the code provided by ChatGPT, I stumbled upon a series of challenges. But the AI guide stood by my side, helping me troubleshoot and refine the code until it was perfect. It was during this process that I encountered a persistent error, and ChatGPT suggested: "We can add more print statements to help identify the issue further." It was at this moment that I realised the tables had turned - the "robot" was using me, a human, as a debugging console.

MK

```
print(f"item['created_at']: {item['created_at']}")
```

Exception has occurred: TypeError
string indices must be integers

I apologize for the confusion. It seems that the error is caused by an incorrect item being passed to the `get_time_since_created()` function. **We can add more print statements to help identify the issue further.**

Please modify the `main()` function to include print statements right before the sorting step, like this:

python

Copy code

```
for idx, item in enumerate(project_duration_data):  
    print(f"Item {idx}: {item}")  
  
project_duration_data.sort(key=lambda x: max(get_time_since_created(x["longe
```

Now, run the script again, and let me know what the output is right before the error. This will help us understand what's causing the issue with the items passed to the `get_time_since_created()` function.

This unexpected role reversal led me to appreciate the power of AI in a new light. ChatGPT had transformed into an interactive console, guiding me through the debugging process with patience and wisdom. Together, we navigated the intricate world of GitLab APIs and Python code, resolving issues and refining solutions. Our collaboration became a story of discovery, learning, and problem-solving.

MK

ok, now it seems that its works. but i am not getting any merge requests or issues. i am thinking that reason for that should be that merge request are connected to groups and not to project



It's possible that the GitLab API is returning merge requests differently for your setup, especially if you have a large number of groups and projects. If merge requests are connected to groups, we can modify the script to fetch merge requests from groups instead of projects.



First, let's modify the ``get_open_issues_or_merge_requests()`` function to handle group requests:

python

Copy code

```
def get_open_issues_or_merge_requests(item_id, resource, is_group=False):
    items = []
    page = 1
    item_type = "groups" if is_group else "projects"

    while True:
        response = requests.get(f"{gitlab_api_base}/{item_type}/{item_id}/{resource}")
        open_items = response.json()

        if not open_items or not isinstance(open_items, list):
            break

        items.extend(open_items)
        page += 1

    return items
```

Now, let's modify the ``main()`` function to fetch groups and their merge requests:

python

Copy code

```
def main():
    projects = get_all_projects()
    groups = get_all_groups()
    project_duration_data = []
```

However, my journey with ChatGPT also revealed the limitations and risks of AI. As powerful and insightful as the AI language model may be, it is still dependent on the context and information provided by the human user. If I had given ChatGPT the wrong context, our collaboration might not have been as successful. While AI has made significant progress, it is essential to remember that it is not perfect. The human touch is still crucial in guiding the AI towards the right path and making the most of its capabilities.

In the end, not only did ChatGPT help me craft the perfect Python script, but it also taught me invaluable lessons about troubleshooting and refining code. As a final request, I asked ChatGPT to assist me in sorting projects by the descending median time to close merge requests. Once again, ChatGPT delivered.



Sure! To sort projects by the descending median time to close merge requests, you can create a list of dictionaries containing the project information and the calculated median times. After processing all projects, you can sort this list and then print the sorted results to the output file. Here's the updated `main()` function:



python

Copy code

```
def main():
    projects = get_all_projects()
    project_medians = []

    for project in projects:
        closed_issues = get_closed_issues_or_merge_requests(project["id"], "
        closed_merge_requests = get_closed_issues_or_merge_requests(project[

        median_time_to_close_issues = calculate_median_time_to_close(closed_
        median_time_to_close_mrs = calculate_median_time_to_close(closed_mer

    project_medians.append({
        "project": project,
        "median_issues": median_time_to_close_issues,
        "median_mrs": median_time_to_close_mrs
    })

    # Sort projects by descending median time to close merge requests
    project_medians.sort(key=lambda x: x["median_mrs"], reverse=True)

    # Open the output file
    with open("output.txt", "w") as output_file:
        for project_data in project_medians:
            project = project_data["project"]
            median_issues = project_data["median_issues"]
            median_mrs = project_data["median_mrs"]

            # Redirect print statements to output_file
            print(f"Project: {project['name']} (ID: {project['id']})", file=
            print(f"Median time to close issues: {median_issues}", file=outp
            print(f"Median time to close merge requests: {median_mrs}", file
            print(file=output_file)

        print("Results sorted and saved to output.txt")

if __name__ == "__main__":
    main()
```

My adventure with ChatGPT was an extraordinary journey of code, collaboration, and AI-powered guidance. It was a reminder that, with the right guidance, even the most complex problems can be conquered.

And so, dear colleagues, I share with you this story of a rainy Sunday, where the lines between human and AI blurred, and I became an interactive console for an AI language model. May your own encounters with ChatGPT be as inspiring and transformative as mine, and remember that the collaboration between human and AI is the key to unlocking their true potential.

Michal Kovacik