**People10 Code Challenge**



This project is designed as a quick exercise to gauge a candidate's understanding of software design and general programming knowledge

**Instructions:**

1. Use any of the PHP frameworks such as Symfony, Laravel etc
2. Please use composer to manage the dependencies
3. Please ensure to implement the project such that it can be easily added as a dependency in any other php project. (We are specifically interested to see your skill of developing a generic library that can be used across multiple projects)
4. Treat it as a "real-world component" that will be added into our system.
5. Please take into account to demonstrate your strengths and not leave your experience and capabilities to question.
6. If possible please include a readme file such that a user with basic systems administration knowledge is able to have your program up and running in under 5 minutes.
7. Your program must be compatible with OSX and Linux.
8. Bonus for writing Unit tests.
9. Super Bonus for incorporating any library which does linting check on the source code to analyse code for potential errors.
10. We encourage you to use software engineering best practices that you recommend and live by in designing your solution.

**Performance considerations**

When thinking about the structure of your database, assume that the most common operations are GET the data, SET the data, UPDATE the data and DELETE the data, all of which are equally common. All these commands should have an expected worst-case of O(log(N)) or better, where N is the total number of variables stored in the database.

Additionally, you should assume that only a small number of values will be changed in a transaction. Your solution should be efficient about how much memory is used by a transaction (i.e. it should not nearly double your program's memory usage).

If you have questions or need clarification on the project, please feel free to ask question to us, and they will be happy to assist you.

**The** ​**Challenge**

Your task is to -

- Create a very simple MySQL database having two tables that is

**employees[ id, emp\_id, epm\_name, ip\_address]** and **employee\_web\_history[ip\_address, url, date]** tables. These table should be created through migration files.

- Create GET, POST, DELETE API end points to get, insert and delete the employee websearch data based on the **ip\_address** as a key. Your code should must have

model for each table, controller to handle the API requests and repository to do operation and format data.

- Create 3 console command to operate these API requests in command line interface. All of the commands are going to be fed to you one line at a time via stdin, and your job is to process the commands and to perform whatever operation the command dictates. All the response will come with json result.

**Data Commands**

Your functionality should accept the following commands:

* **SET empdata [emp\_id] [emp\_name] [ip\_address]** : Insert the employee details to employee table with data emp\_id, emp\_name, ip\_address.
* **GET empdata [ip\_address]** : Get the employee details having the ip\_address
* **UNSET empdata [ip\_address]** : Soft delete the data having the passed ip\_address
* **SET empwebhistory [ip\_address] [url]​** : It will first check if the ip address is assigned to any employee or not if the ip address is there then it will insert the url variable [url] to the mapped ip\_address [ip\_address], other with it will throw error.
* **GET empwebhistory [ip\_address]** ​: Print out the employee details with his web search history stored under the variable [ip\_address]. Print NULL if that ip\_address doesn’t have any data
* **UNSET empwebhistory [ip\_address]​** :Delete all the web search history data mapped with ip\_address.
* **END​**:Exit the program.

**Examples**

So here is a sample input:

1)

SET empdata 1 ‘Jack Petter’ ‘192.168.10.10’

GET empdata ‘192.168.10.10’

UNSET empdata ‘192.168.10.10’

GET empdata ‘192.168.10.10’

SET empwebhistory 192.168.10.10 ‘http://google.com’

GET empwebhistory 192.168.10.10

UNSET empwebhistory 192.168.10.10

GET empwebhistory 192.168.10.10

END

And its corresponding output:

"employee": {

"id": 1

"empId": "1",

"empName": "Jack Petter",

"empIpAddress": "191.168.10.10"

}

NULL

Resource not found

NULL

Resource not found

NULL

2)

SET empdata 1 ‘Jack Petter’ ‘192.168.10.10’

GET empdata ‘192.168.10.10’

SET empwebhistory 192.168.10.10 ‘[http://google.com](http://google.com/)’

SET empwebhistory 192.168.10.10 ‘http://facebook.com’

GET empwebhistory 192.168.10.10

UNSET empwebhistory 192.168.10.10

GET empwebhistory 192.168.10.10

END

And its corresponding output:

"employee": {

"id": 1

"empid": "1",

"empName": "Jack Petter",

"empIpAddress": "191.168.10.10"

}

“employeewebhistory”: {

"id": 1

"empIpAddress": "191.168.10.10",

"urls": {

“url”: "[http://google.com](http://google.com/)",

“url”: “http://facebook.com”

}

}

NULL