Description

CMDB Dashboard is a backstage management system which represents CMDB data in a visulaized way and allows users to implement their customized queries.

Server Environment

Apache + PHP + MySQL

Technology Stack

HTML + CSS + JavaScript + JQuery + PHP + MySQL

Dependency

- light bootstrap dashboard
- animate
- echarts

Project Structure

```
|-- assets
     |-- css
        |-- animate.min.css
         |-- light-bootstrap-dashboard.css
         |-- pe-icon-7-stroke.css
         |-- style.css
     |-- fonts
         |--Pe-icon-7-stroke.eot
         |--Pe-icon-7-stroke.svg
         |--Pe-icon-7-stroke.ttf
         |--Pe-icon-7-stroke.woff
     |-- js
          |--echarts.js
          |--jquery.table2excel.min.js
          |--light-bootstrap-dashboard.js
          |--load_application_charts.js
          |--load_backup_charts.js
```

```
            |--load_basic_data.js
            |--load_build_charts.js
            |--load_networking_charts.js
            |--load_reboot_charts.js
            |--load_report_charts.js
            |--load_retire_charts.js
            |--load_server_charts.js
|-- backend
    |--application
        |--app_lookup.php
        |--app_sla.php
        |--search_app.php
    |--backup
        |--backup_baremetal.php
        |--backup_frequency.php
        |--backup_lookup.php
        |--backup_retention.php
        |--search_backup.php
    |--build
        |--build_daily.php
        |--build_incident.php
        |--build_user.php
        |--build_week_month_year.php
        |--search_build.php
    |--custom
        |--delete_query.php
        |--report_all.php
        |--report_category.php
        |--report_fetch.php
        |--report_single.php
        |--update_query.php
    |--networking
        |--networking_ip.php
        |--networking_lookup.php
        |--networking_owners.php
        |--networking_zone.php
        |--search_networking.php
    |--reboot
        |--reboot_enabled.php
        |--reboot_schedule.php
        |--reboot_time.php
        |--reboot_weeks.php
        |--search_reboot.php
    |--retire
        |--retire_daily.php
        |--retire_incident.php
        |--retire_ip.php
        |--retire_user.php
        |--retire_week_month_year.php
```

```
        |--search_retire.php
    |--server
        |--search_server.php
        |--server_app.php
        |--server_backup.php
        |--server_datacenter.php
        |--server_info.php
        |--server_managed.php
        |--server_os.php
        |--server_owner.php
        |--server_status.php
        |--server_type.php
    |--alert.php
    |--basic_info.php
    |--connection.php
|-- application.php
|-- backup.php
|-- build.php
|-- custom.php
|-- dashboard.php
|-- index.php
|-- networking.php
|-- reboot.php
|-- retire.php
|-- server.php
|-- README.md
```

How to add a new chart later?

When adding a new chart in this project, we need to consider these following 3 questions.

> 1. Where does the data come from?
> 2. How to organize the data into echarts?
> 3. How to represent the echats in the website?

I will use **Server Owner chart** as an example to introduce how to add a new chart.

> **1. Where does the data come from?**

1.1 Create a new PHP file in the following folder. **ProjectName->backend->server->server_owner.php**

1.2 Fetch data from MySQL database using PHP coding.

```php
<?php

//Connect to the database
include("../connection.php");

//Here we group the owner with relative number
$query = "SELECT count(team.name) as count, team.name as name
FROM server,team
WHERE server.owner_idx=team.team_idx
GROUP BY team.name";

//Create an array to store data later
$data = array();

//Execute the query in the database
$resultset= mysqli_query($conn, $query);

//Create a class  to store data later
class ServerOwner{
    public $owner;
    public $counts;
}

//Read the data from database
while($row = mysqli_fetch_array($resultset)) {
    $serverOwner = new ServerOwner();
    $serverOwner->owner = $row['name'];
    $serverOwner->counts = $row['count'];
    $data[] = $serverOwner;
}

//Close the database
mysqli_close($conn);

//Transform the data into JSON and export the data to the front-
echo json_encode($data);
?>
```

So far, you have already gotten the JSON based data in the backend. You can check the data by simply inputting the following address. http://HostName/ProjectName/backend/server/server_owner.php

> **2. How to organize the data into echarts?**

2.1 Go to folder **ProjectName->assets->js**, and open file **load_server_charts.js**

2.2 You should fetch the data generated in **STEP 1.2** through Ajax by following coding.

```
//These variables are based on what you export in the PHP fi
var ownerNames = [], counts = [];

function getServerOwner() {
    $.ajax({
        type: "post",
        async: false,
        url: "backend/server/server_owner.php",
        data: {},
        dataType: "json",
        success: function (result) {
        //Here, the result is the JSON  data exported by PHP
            if (result) {
                for (var i = 0; i < result.length; i++) {
                    ownerNames.push(result[i].owner);
                    counts.push(result[i].counts);
                }
            }
        },
        error: function (errmsg) {
            alert("Ajax wrong!" + errmsg);
        }
    });
    return ownerNames, counts;
}

getServerOwner();
```

2.3 Initialize the echarts through following coding

```
var server_owner_charts = echarts.init(document.getElementBy
//Here, the container_server_owner is the id of an HTML <div
```

2.4 Set the echart's parameter

```
var option_owner = {
    title: {
        text: 'Server Owner',
        x: 'center'
    },
    tooltip: {
```

```
            trigger: 'axis',
            axisPointer: {
                type: 'shadow'
            }
        },
        legend: {
            data: ['Amout'],
            x: 'left'
        },
        grid: {
            left: '3%',
            right: '4%',
            bottom: '3%',
            containLabel: true
        },
        xAxis: {
            type: 'category',
            data: ownerNames,//The variable ownerNames comes fro
            axisLabel: {
                rotate: 70,
                interval: 0,
                fontSize: 10
            },
        },

        yAxis: {
            type: 'value',
            boundaryGap: [0, 0.01]
        },
        series: [{
                "name": "Amout",
                "type": "bar",// The chart type is bar
                "data": counts,//The variable counts comes from
            }
        ]
    };
```

2.5 Bundle the echart parameter with the initialized echart

```
    server_owner_charts.setOption(option_owner);
```

2.6 Make the echarts clickable with detail information

```
       server_owner_charts.on('click', function (params) {
        var owner = params.name;
```

```
        $.ajax({
            url: "backend/server/search_server.php",// You don't
            data: { owner: owner},
            type: 'POST',
            success: function (result) {
                if (result) {
                    $('#table-frame').fadeIn();
                    $('#table_title').html('Server owner: ' + ow
                    $("#myTable").html(result);
                }
            }
        })
    });
```

So far, you have already gotten all the codings about how to organize the data into echarts. If you want to know more about how echarts works, **HERE** is the offcial tutorial from its website.

> **3. How to represent the echats in the website?**

3.1 Go to **root folder**, and open file **server.php**

3.2 In this file, the main function is to export the HTML coding of the server chart. The HTML structure is based on Bootstrap grid system. **HERE** is the turorial about how Bootstrap grid system works.

3.3 To show the echart in HTML, you should have to create a new div in a right row with right size of column and an id. Here, the id is what you should use in **STEP 2.3**.

```
<div class="row">
    <div class="col-md-12">
        <div class="card height-4" id="container_server_owner"><
    </div>
</div>
```

Then, all works done. Cheers!