PHP Documentacion

Curso: 2024/2025

Autores

Javier Zurita Torres

ÍNDICE

PHP Documentacion..... 1
Relación de archivos, ubicación y breve descripción de su función..... 3
Secciones de código relevante..... 3
Controlador: StatsController.php..... 3
Consola: console.php..... 4
Limitaciones de la implementación..... 4
Fuentes y repositorios..... 4
Dependencias y despliegue..... 5
Dependencias:..... 5
Despliegue:..... 5
Notas técnicas de soporte..... 5
Configuración del scheduler en Windows:..... 5
Resumen:..... 5

Parte 1: Documentación técnica del módulo
El presente documento describe de forma técnica el módulo implementado en el proyecto, desarrollado con el framework Laravel. Se detalla la ubicación y función de los archivos esenciales, se destacan las secciones de código relevantes, se comentan las limitaciones detectadas en la implementación, se indican las dependencias necesarias para la ejecución y despliegue, y se incluyen notas técnicas de soporte.

Relación de archivos, ubicación y breve descripción de su función

En este proyecto, la estructura básica sigue la preinstalada al crear un proyecto Laravel con PHPUnit y MySQL (sin migraciones previas). Los archivos directamente relacionados con el módulo son:

1. **StatsController.php**
    - **Ubicación**: `app/Http/Controllers/StatsController.php`
    - **Función**: Controlador generado con `php artisan make:controller StatsController --resource`. Contiene las funciones para recolectar, procesar y almacenar datos meteorológicos, incluyendo conversiones de codificación y coordenadas.
2. **console.php**
    - **Ubicación**: `routes/console.php`
    - **Función**: Archivo de rutas para comandos de consola. Programa tareas periódicas (ejecución de recolecta y procesamiento de datos) mediante el scheduler de Laravel.

Secciones de código relevante

Controlador: StatsController.php

- **Función `recolectaInv(int $cantidad = 5)`**
    - **Conversión de codificación**:
      php
      Copy
      mb_convert_encoding($datas, "UTF-8", "ISO-8859-1");
      Convierte datos JSON de ISO-8859-1 a UTF-8 para evitar errores de codificación.

Conversión de coordenadas geográficas:
php
Copy
dmsToDecimal($datajson[$i]['latitud']);

- dmsToDecimal($datajson[$i]['longitud']);
  Transforma coordenadas en formato DMS (grados, minutos, segundos) a decimal para almacenamiento en la base de datos.

Consola: console.php

Retry en tareas programadas:
php
Copy
retry(3, function () {

$controller = new StatsController();

$controller->almacenaStat();

- }, 5000);
  Reintenta hasta 3 veces la ejecución de `almacenaStat()` en caso de fallos (por ejemplo, errores de conexión con la API de AEMET).

Limitaciones de la implementación

1. **Configuración manual del scheduler**:
   - Las tareas programadas requieren configuración externa del sistema operativo (ej. cron en Linux o Task Scheduler en Windows).
   - Cualquier cambio en las tareas implica reconfigurar manualmente el sistema.
2. **Procesamiento de datos**:
   - Los datos se validan recorriendo toda la base de datos, lo que puede ralentizar el tiempo de ejecución. Se explora optimizar con índices o caché.

Fuentes y repositorios

- **Documentación oficial de Laravel**: Base para el uso de schedules, controladores y comandos.
- **Repositorio GitHub**: Único repositorio con control de versiones, desplegado mediante GitHub Desktop.

Dependencias y despliegue

Dependencias:

- **PHP**: Versión 8.0 o superior.
- **Laravel**: Versión 10.x/11.x.
- **MySQL**: Base de datos para almacenamiento.
- **Composer**: Gestión de paquetes.
- **Extensiones PHP**: `mbstring`, `openssl`, `pdo_mysql`.

Despliegue:

- **Configuración del scheduler**:
  - **Linux/Unix**:
    bash
    Copy
    * * * * * cd /ruta/proyecto && php artisan schedule:run >> /dev/null 2>&1
  - **Windows**:
    - Crear tarea en el **Programador de Tareas** con el comando:
      bash
      Copy
      php C:\ruta\proyecto\artisan schedule:run

Notas técnicas de soporte

Configuración del scheduler en Windows:

1. Abrir el **Programador de Tareas**.
2. Crear una nueva tarea:
   - **Disparador**: Ejecutar cada minuto.
   - **Acción**:
     - **Programa/Script**: `php.exe` (ruta completa si no está en PATH).

        - **Argumentos**: `C:\ruta\proyecto\artisan schedule:run`.
3. Asegurar que la tarea se ejecute incluso sin sesión de usuario activa.

Resumen:

- **Linux/Unix**: Usar cron para ejecutar `php artisan schedule:run` cada minuto.
- **Windows**: Configurar tarea programada con el comando completo.