freepikcompany

Backoffice Developer Coding Task

Video games listing API

Se requiere la realización de una pequeña api REST que devuelva un listado de videojuegos obtenido desde un SGBD MySQL.

Los parámetros de conexión a la base de datos son:

Nombre de la base de datos	`good_old_games`
Usuario del SGBD	`root`
Password del SGBD	`root`
Nombre del *host* del SGBD	`db`
Puerto por el que acceder al SGBD	`3306`

Petición

La aplicación comprenderá tres *end points*:

End point	Descripción
`/companies/{id}`	Recepcionará un ID de la compañía desarrolladora de la que se desea obtener los videojuegos.
`/systems/{id}`	Recepcionará un ID de la plataforma para la que fue desarrollada los videojuego que se quieren obtener.
`/games`	-

Todos los *end points* reciben, además, el parámetro **page**, siendo un número entero usado para la paginación de los resultados.

Respuesta

Como resultado, se devolverá un JSON que contendrá una lista con los videojuegos solicitados y paginados de tres en tres, además de unos *tags* asociados que se generarán según estos criterios:

- Basado en localización: Añadir a la salida el tag "Nipón" si la compañía del sistema está ubicada en japón.
- Basado en la fecha de publicación: Añadir a la salida el tag "Vintage" si el videojuego fue lanzado hace 20 años o más. Añadir el tag "Oldie but Goldie" si fue lanzado hace 30 años o más.
- Basado en el género: Añadir a la salida el tag "Machacabotones" si el género del videojuego es lucha o beat 'em up.

La salida por pantalla debería ser, por tanto:

```
[
    {
        "id":14,
        "title":"Soul Calibur",
        "type":"Lucha",
        "released_on":"1999-08-05",
        "company":"Namco",
        "system":"Dreamcast",
        "tags": [
            "Nipón",
            "Vintage",
            "Machacabotones"
        ]
    }
]
```

En caso de error, devolver una respuesta con la información de valor que estimes oportuna.

Pruebas

La aplicación deberá estar cubierta por pruebas unitarias y funcionales.

Requisitos no funcionales

Aunque el ámbito de la aplicación es limitado, esta deberá ser desarrollada como si se tratara de un proyecto complejo. Se valorará el uso y la correcta configuración de un sistema de inyección de dependencias.

Respecto a las pruebas, se valorará el uso de una librería de *mocks*, como Mockery[1] o Prophecy[2].

Restricciones

- Usa Slim Framework 3 o superior.
- Usa PHP 7.2 o superior.
- Usa PHPUnit Respecto a los tests, se va
- Puedes enviar la url de un repositorio en GitHub de tu prueba.

Consideraciones Extras

- Puedes elegir la arquitectura que desees, pero los conocimientos sobre *Clean Architectures* (Hexagonal / Ports & Adapters, DDD, CQRS) serán valorados muy positivamente.
- Un conocimiento profundo de principios como SOLID, YAGNI o KISS serán valorados positivamente.
- Conocimientos DEVOPS como pueden ser un entorno de desarrollo basado en docker o ficheros de configuración de despliegue automático (GitHub Actions, Bitbucket Pipelines o Jenkins) serán valorados positivamente.
- Es importante un equilibrio entre la calidad del código y el tiempo de entrega: entregar a tiempo un código que no funciona o entregar un código perfecto fuera de plazo podrían evaluarse negativamente.
- No es obligatorio entregar un repositorio en GitHub, pero se valorará positivamente la documentación de los *commits* del proyecto y el uso que hace el desarrollador de las herramientas de control de versiones.
- El uso eficiente e inteligente de librerías de terceros se valorará positivamente.
- Más allá de los requisitos de esta prueba, queremos ver de qué eres capaz, siéntete libre de mostrarnos tu potencial real.

[1] "mockery/mockery: Mockery is a simple yet flexible ... - GitHub." https://github.com/mockery/mockery.
[2] "phpspec/prophecy: Highly opinionated mocking ... - GitHub." https://github.com/phpspec/prophecy.