

1º Realiza un servicio con las siguientes rutas y funcionalidades dentro del mismo servicio:

- Escribe una ruta que simule el mecanismo de devolución de monedas de una máquina expendedora. El sistema admitirá una cantidad de dinero y calculará la cantidad de monedas necesarias para dar esa cantidad. Por ejemplo, 3,47 € serían 1 moneda de 2€, 1 de 1€, 2 de 20cts, 1 de 5cts y 1 de 2 cts.
- Crea un servicio que recoja del cliente la fecha actual y la fecha de nacimiento de una persona; el programa determinará la edad. Si la fecha actual no se recibe del cliente se usará la fecha del sistema para el cálculo.
- Realiza un servicio que calcule la suma de los dígitos de un número. Si el servicio recibe dos números determinará cual es el que la suma de sus dígitos es mayor.

2º Realiza un servicio con las siguientes rutas y funcionalidades dentro del mismo servicio:

- Dada una cadena, y un carácter, verificar cuántas veces se repite el carácter en la cadena, por ejemplo:
Entrada: cad = "cAsa blanCA", car = 'a'
Salida: El carácter 'a' se repite 4 veces
- La clave del César.
Un grupo de inteligencia militar desea codificar los mensajes secretos de tal forma que no puedan ser interpretados con una lectura directa, para lo cual han establecido las siguientes reglas (ya inventadas por Julio César):
 - o Todo mensaje debe tener sus letras en mayúsculas.
 - o Reemplazar cada letra por la que sigue según abecedario, excepto Z que se deberá reemplazar con la letra A.
 - o Reemplazar cada dígito encontrado por el siguiente número excepto el 9 que deberá ser reemplazado por el 0.
- Escribe una ruta que reciba una frase (cad) y devuelva cuántas palabras contiene.
Ejemplo:
Entrada: "Laravel es divertido"
Salida: { "palabras": 3 }
- Crea un servicio que reciba una cadena (cad) y determine si es un palíndromo (se lee igual de izquierda a derecha y de derecha a izquierda, ignorando espacios y mayúsculas). Debe devolver un JSON con { "resultado": "si es un palindromo" } o { "resultado": "no es un palindromo" }. Realiza el control de errores de modo que la cadena admitida no contenga números y solo letras sin caracteres especiales.
- Crea una ruta que reciba dos cadenas de texto (parámetros cad1 y cad2). El servicio debe determinar si ambas cadenas son anagramas y devolver el resultado en formato JSON.
Un anagrama es una palabra o frase que resulta de reorganizar las letras de otra, utilizando todas las letras exactamente una vez.
Por ejemplo:
 - o "amor" y "roma" son anagramas.
 - o "listen" y "silent" son anagramas.
 - o "perro" y "gato" no son anagramas.



SERVIDOR 2º CFGS DAW Ejercicios iniciales Laravel



Página 2 de 4

3º Juego de las parejas.

Realizaremos el juego de las parejas. Se inicia un vector de n casillas (siendo n un número par) y se colocan al azar parejas de números. Ese panel se oculta al jugador al que se le mostrará un panel vacío del que irá destapando de 2 en 2. Si los números destapados coinciden se quedan visibles si no se muestran un segundo y luego se ocultan. El jugador tratará de recordar qué números eran para encontrar a su pareja. Este es el juego, realiza las rutas necesarias para dar soporte a un cliente que implementara dicho programa.

Piensa en los verbos necesarios, como pasar la información y datos (JSON y códigos más correctos) devueltos. Cuando sea posible trata de implementar persistencia usando tanto Query Builder como Eloquent. Lógicamente tendrás que diseñar previamente la BD que recoja la información necesaria.

4º Estarword.

Se trata de diseñar el backend de la gestión de la flota, pilotos y mantenimiento de naves del universo Star Wars, usando Eloquent ORM y relaciones entre modelos. Los datos los puedes sacar de la API: <https://swapi.dev/api/>

Tendremos **planetas** que tendrá asociado varias **naves**. De los **planetas** nos interesa saber: **nombre, período de rotación, población y clima** y de las **naves**: **nombre, modelo, tripulación, pasajeros y clase de nave**. Las naves pertenecen a un planeta y puede tener varios mantenimientos y varios pilotos asociados. De los mantenimientos: **id, idnave, fecha, descripción y coste**. De los **pilotos** tendremos también que guardar información: **nombre, altura, año de nacimiento y género**.

Tendremos que gestionar que las naves son pilotadas por pilotos; necesitaremos una tabla pivote que tendrá información del piloto, de la nave, de la fecha en la que el piloto está asociado y la fecha de fin de asociación.

El cliente necesitará la siguiente información/funcionalidad:

- CRUD de naves. Por ahora abierto, pero que luego protegeremos.
- Listado de toda la información almacenada. Listados generales y búsquedas por id.
- Asignar/Desasignar un piloto a una nave. Con control de errores.
- Listar todas las naves que no tienen piloto.
- Listar todos los pilotos asignados a naves (histórico, no tienen por qué estar asignados actualmente).
- Lo mismo que el punto anterior pero solo mostrar los pilotos que actualmente están asociados a naves y las naves.
- Registrar un mantenimiento.
- Listar mantenimientos puntuales.
- Listar mantenimientos de naves entre dos fechas.

Nuestra aplicación ya gestiona todo eso, vamos a añadirle un sistema de gestión de usuarios para acceder a la aplicación. Vamos a manejar tres **roles**:

- admin -> podrá hacer todo. Todas las rutas anteriores y la gestión de los usuarios que acceden al sistema. Podrán conceder y revocar roles también, claro.
- gestor -> podrán gestionar los mantenimientos (altas y bajas de los mismos), asociar/quitar pilotos y naves y acceder a todos los listados del sistema (menos el de los usuarios, la gestión íntegra de los usuarios pertenece al admin).



SERVIDOR 2º CFGS DAW Ejercicios iniciales Laravel



Página 3 de 4

- usuario -> podrá solo acceder a los listados y consultar todo (menos los usuarios) pero no podrá modificar nada.

Añade varios **validator** para que el almacenamiento de los datos de las naves, pilotos, usuarios, mantenimientos, etc... sea coherente.

Vamos a introducir **imágenes** a nuestra app. Cuando registramos pilotos tendrán asociada una imagen genérica (elige la que te guste), esta imagen estará almacenada de forma local. El usuario podrá subir su foto de perfil personalizada, que se almacenará en Cloudinary. Realiza un cliente que me permita probar subida y descarga/muestra de foto.

Tests.

Realiza un test de cada tipo:

- Unitario: para realizar este test diseña una función que calcule los días de mantenimiento y le aplique un coste base de 100 € por día. Esta función calculará los costes de un mantenimiento acabado y debe tener su test correspondiente.
- Funcional: que testee varias funcionalidades: registro de naves, borrado de pilotos y modificación de mantenimientos.
- Integración: que testee toda la funcionalidad comprendida en: registro de piloto, registro de nave y asignación de piloto a nave.

5º Juego del Lingo.

El juego podía considerarse una mezcla entre el Bingo y el Master Mind. Las parejas de concursantes tenían que adivinar palabras de 5 letras, en un máximo de 5 intentos. Se les concedía la letra inicial y cuando decían una palabra, se les indicaba si alguna de las letras utilizadas formaba parte de la palabra oculta. Si la letra estaba en el sitio exacto, se recuadraba de rosa, y si la letra estaba en la palabra, pero no en el sitio exacto, se circulaba de amarillo.

Ejemplo:

Palabra buscada y oculta: hueso.

Pieistas: ----

Intento: cosas

Pieistas: -**_*

Intento: terco

Pieistas: -*--o

Intento: rueda

Pieistas: -ue-*

Intento: suelo

Pieistas: *ue-o

Intento: hueso

Has acertado.

Piensa en los verbos necesarios, como pasar la información y datos (JSON y códigos más correctos) devueltos. Cuando sea posible trata de implementar persistencia usando tanto Query Builder como Eloquent. Lógicamente tendrás que diseñar previamente la BD que recoja la información necesaria.



CIFF VIRGEN DE GRACIA

SERVIDOR 2º CFGS DAW Ejercicios iniciales Laravel



Página 4 de 4

Idea alguna forma de crear un repositorio de palabras.

6º Piedra, papel, tijera.

Realiza un servicio en Laravel que permita jugar contra la máquina a piedra, papel, tijera.

Se debe jugar al mejor de cinco partidas (el que gane 3, gana). Cada usuario solo puede tener una partida abierta, hasta que no se acabe no se permite comenzar otra.

Los servicios proporcionados serán:

- Comenzar partida.
- Información del usuario (solo un usuario puede ver sus datos).
- Realizar tirada.

Se debe realizar un control de errores adecuado en parámetros.

Para la gestión de usuarios se realizará un servicio aparte que será gestionado por un administrador.

A modo de recordatorio:

- Piedra le gana a la tijera.
- Tijera gana al papel.
- Papel le gana a la piedra.