Test de Conocimientos Generales para Desarrollador Fullstack Junior

Sección 1: Symfony

1. **Pregunta de Configuración:**

- Describe los pasos básicos para levantar un proyecto en Symfony.

Para levantar un proyecto en symfony, debemos tener symfony cli instalado, composer, scoop y el proyecto debe estar creado, hay distinciones entre crear un proyecto monolito (webapp) y una api, microservicio o aplicación de consola, adicionalmente puedes verificar que se instalado symfony correctamente con el comando

- Crear un proyecto monolito (webapp)

  symfony new my_project_directory --version="7.1.*" --webapp

- Crear un proyecto Api, console, microservicio

  symfony new my_project_directory --version="7.1.*"

- Entrar a la carpeta del proyecto a través de la consola y ejecutar el comando

  symfony server: start

2. **Pregunta de Código:**

- Crea un controlador en Symfony que maneje una ruta /hello/{name} y devuelva un saludo personalizado. Además, si el nombre no se proporciona, debe devolver un saludo predeterminado "Hello World". (opcional) Implementa también una prueba unitaria para verificar que la ruta devuelve el saludo correcto.

Se crea dentro de la carpeta Controller el controlador WelcomeController, con el mappeo para las rutas con la variable name ('/hello/{ name }') y sin la variable name('/hello'), así mismo se creó dentro de la carpeta tests/Controller un test para los paths '/hello' y '/hello/{name}'

3. **Pregunta Teórica:**

- Explica la arquitectura de Symfony y cómo se organiza un proyecto típico en términos de carpetas y archivos.

Symfony crea un scaffolding con la estructura del proyecto, en la raíz del proyecto podemos encontrar los archivos .env, donde nos podemos encontrar diferentes configuraciones como son el ambiente(APP_ENV=dev) en el que nos encontramos, la clave secreta de aplicación(APP_SECRET=) así como configuraciones para el acceso a la base de datos para diferentes gestores (mysql, postgres, sqlite), en la raíz del proyecto también podemos encontrar el archivo composer.json donde se registran las versiones de los módulos que descargamos así como la versión de symfony que estamos usando("symfony/dotenv": "7.1.*"), también nos podemos encontrar una de las carpetas más importantes src(source) donde se guardaran los códigos que creamos en las carpetas Controller, Entity y Repository las cuales se crean por defecto junto con la inicialización del proyecto, y a medida que se van generando entidades y controladores se van poblando, también

nos podemos encontrar los templates(plantillas) que serían las vistas de la aplicación, symfony usa twig como motor de plantillas, otra de las carpetas es las migrations(migraciones), que son las actualizaciones que hacemos del modelo con la base de datos, en la carpeta config nos encontramos las configuraciones de los bundle (paquetes) que instalamos si así lo requieren el archivo route.yml que en las nuevas versiones está configurado apuntando donde están los controladores para generar las rutas y el archivos bundles.php que todos los bundles que se instalen los va a registrar y a cargar, la carpeta assets, es donde se guardaran los archivos javascript y css, la carpeta test es donde se guardaran los test creados

4. **Pregunta de Código:**

- Escribe un servicio en Symfony que se inyecta en un controlador y realiza una operación matemática básica (por ejemplo, sumar dos números). ¿Qué configuraciones son necesarias para poder usarlo? (opcional) Implementa también una prueba unitaria para verificar el correcto funcionamiento del servicio.

Para inyectar un servicio a un controlador, la configuración básica viene hecha por defecto en el archivo service.yml dentro de la carpeta config. Para la realización de esta operación, se creó el controlador MathOperationController que hereda de la clase AbstractController dentro de la carpeta Controller, en el constructor del controlador se le inyecto el servicio y en el cuerpo del constructor se inicializo la variable local $mathOperationService para después ser utilizada llamando al método del servicio que realiza la operación.

Esta es la ruta del controlador para hacer la operación, después de la palabra suma se le pone los números que deseas sumar separándolos por “/”

- http://127.0.0.1:8000/suma/12/9

5. **Pregunta de Código:**

- Muestra cómo crear un formulario en Symfony para una entidad User con campos username y email.

Para crear un formulario para la entidad User la entidad debe estar creada con sus respectivas propiedades y sus métodos getter y setter.

Creamos un nuevo formulario se puede hacer manual o con el comando

- php bin/console make:form UserType

El asistente le permite ingresar la entidad de la cual desea crear el formulario entonces generará un formulario llamado UserType a partir de la clase User en la carpeta Form.

6. **Pregunta Teórica:**

- Explica el concepto de Dependency Injection (DI) en Symfony y cómo se configura.

La inyección de dependencias es un patrón de diseño que se utiliza ampliamente en Symfony. En Symfony, la inyección de dependencias se refiere a la técnica de proporcionar a un objeto las dependencias que necesita para realizar su trabajo, en lugar de que el objeto mismo las cree. En el contexto de Symfony, las dependencias suelen ser servicios. Al inyectar dependencias en un objeto, estás desacoplando el objeto de las implementaciones concretas de esas dependencias, lo que hace que tu código sea más flexible, mantenible y fácil de probar.

La inyección de dependencias en symfony se configura principalmente en el archivo services.yml. Este archivo se encuentra generalmente en el directorio config del proyecto Symfony y se utiliza para definir los servicios de tu aplicación y cómo deben ser instanciados y configurados.

7. **Pregunta de Código:**

- Escribe una consulta Doctrine en Symfony para obtener todos los registros de una entidad Product donde el precio sea mayor a 100.

Para un mejor mantenimiento del código y mayor escalabilidad y seguridad, se debe crear una clase de repositorio en la cual se creara las consultas de la base de datos a través de un ORM o consultas nativas de sql, creamos la clase Product y ProductRepository que extiende de ServiceEntityRepository

```
public function findByPrice(): array
  {
       return $this->createQueryBuilder('p')
          ->where('p.precio > :price')
          ->setParameter('price', 100)
          ->getQuery()
          ->getResult();
  }
```

8. **Pregunta Teórica:**

- ¿Qué es el Event Dispatcher en Symfony y para qué se utiliza?

El Event Dispatcher en Symfony es un componente que permite la gestión y despacho de eventos dentro de una aplicación Symfony. Se utiliza para implementar el patrón de diseño de observador y permite la comunicación entre diferentes partes de la aplicación de forma desacoplada.

9. **Pregunta de Código:**

- Crea un validador personalizado en Symfony para asegurar que el campo email de una entidad User no pertenece a un dominio específico (por ejemplo, "example.com"). Muestra cómo configurar este validador y cómo sería utilizado en la entidad User.

Se debe crear una clase para el validador personalizado en este caso **EmailDomainValidator** que debe implementar la interfaz **ConstraintValidatorInterface,** luego se necesitas crear la restricción

que utilizará tu validador personalizado **EmailDomain**, se necesita configurar el servicio para el validador personalizado en services.yml

```
services:
        App\Validator\Constraints\EmailDomainValidator:
                    tags:
                  - validator.constraint_validator
```

Por ultimo utilizar este validador en la entidad User mediante la anotación @ CustomAssert en el campo email.

```
#[Assert\Email()]
#[CustomAssert\EmailDomain()]
private ?string $email = null;
```

Para esta tarea se creó un directorio Validador y Contraints, donde se crearon las clases **EmailDomainValidator** asi como **EmailDomain**

10. **Pregunta de Código:**

- Implementa un Event Subscriber en Symfony que escuche el evento kernel.request y registre en un archivo de log cada visita a cualquier página de la aplicación. Asegúrate de configurar el servicio correctamente para que el suscriptor se registre con el evento.

Para implementar un Event Subscriber en Symfony que escuche el evento kernel.request y registre cada visita a cualquier página de la aplicación en un archivo de log, se debe crear una clase EventSubscriber que implemente la interfaz EventSubscriberInterface, se le inyecta un logger en el constructor que escriba cada acceso el evento kernel.request, cuando se vaya a escribir en el log, luego hay que configurar en el archivo services.yml el servicio para el EventSubscriber agregándole como argumento el logger

```
App\EventSubscriber\RequestSubscriber:
    arguments: ['@logger']
    tags:
      - { name: 'kernel.event_subscriber' }
```

Para la realización de esta tarea se creó un directorio EventSubscriber y dentro una clase con el nombre RequestSubscriber.php que implementa la interfaz EventSubscriberInterface así como crear la configuración en el archivo services.yml

Sección 2: JavaScript

1. **Pregunta Teórica:**

- Explica la diferencia entre var, let y const en JavaScript.

En JavaScript, var, let y const son formas de declarar variables, pero difieren en cuanto a su alcance y mutabilidad:

Las variables declaradas con var tienen un alcance de función o global, dependiendo de dónde se declaren, si se declara una variable con var dentro de una función, su alcance se limita a esa función, las variables var pueden ser redeclaradas y reasignadas en cualquier lugar dentro de su alcance.

Let fue introducido en ECMAScript 6 (ES6), las variables declaradas con let tienen un alcance de bloque, lo que significa que su alcance se limita al bloque de código más cercano entre {} donde fueron declaradas, las variables let pueden ser reasignadas, pero no pueden ser redeclaradas en el mismo ámbito.

Const también fue introducido en ECMAScript 6 (ES6) y se utiliza para declarar constantes. A diferencia de let, las variables declaradas con const deben ser inicializadas con un valor al momento de la declaración y este valor no puede ser cambiado posteriormente, las variables const no pueden ser redeclaradas ni reasignadas.

2. **Pregunta de Código:**

- Escribe una función en JavaScript que invierta una cadena de texto.

```
function invertirCadena(cadena) {
  return cadena.split('').reverse().join('');
}
```

3. **Pregunta Teórica:**

- ¿Qué es el Event Loop en JavaScript y cómo funciona?

El Event Loop (bucle de eventos) es un concepto fundamental en JavaScript que controla la ejecución de código asíncrono y maneja los eventos en el navegador. Su función principal es garantizar que el código asíncrono se ejecute de manera ordenada y eficiente. El Event Loop opera continuamente, revisando si hay tareas que deben ejecutarse en la cola de tareas.

4. **Pregunta de Código:**

- Escribe un script en JavaScript que filtre los números pares de un array de números y los muestre en la consola.

```
function filtrarNumerosPares(numeros) {
  // Utilizamos el método filter para crear un nuevo array con los números pares
```

```
  var numerosPares = numeros.filter(function(numero) {
    return numero % 2 === 0; // Filtramos los números pares
  });
  return numerosPares;
}
```

5. **Pregunta Teórica:**

- ¿Qué es el DOM y cómo se manipula usando JavaScript?

El DOM (Document Object Model), es una interfaz de programación para documentos HTML y XML. El DOM organiza los elementos HTML en una estructura de árbol, donde cada elemento del documento es representado como un nodo en el árbol. Estos nodos pueden ser elementos HTML, atributos, texto, comentarios, etc. Los nodos se interconectan entre sí, reflejando la estructura del documento.

Se manipula accediendo desde javascript al “document”, donde existen diferentes métodos de manipulación como es el getElementById

// Seleccionar el elemento por su id
var elemento = document.getElementById("miElemento");

// Cambiar el contenido del elemento
elemento.innerHTML = "¡Hola JavaScript!";

6. **Pregunta de Código:**

- Escribe un código en JavaScript que añada un event listener a un botón con el id #myButton para mostrar una alerta con el mensaje "Hello World" al hacer clic.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Event Listener</title>
</head>
<body>

<button id="myButton">Haz clic aquí</button>

<script>
  // Seleccionar el botón por su id
  var boton = document.getElementById("myButton");

  // Agregar un event listener para el evento "click"
  boton.addEventListener("click", function() {
```

```
  alert("Hello World");
 });
</script>

</body>
</html>
```

7. **Pregunta Teórica:**

- Explica qué es una Promesa en JavaScript y proporciona un ejemplo básico.

En JavaScript, una Promesa es un objeto que representa la finalización (o el fracaso) eventual de una operación asíncrona. Una vez que una promesa se resuelve o se rechaza, se dice que está resuelta. Después de que una promesa se resuelve o se rechaza, no puede cambiar de estado.

```
var miPromesa = new Promise(function(resolve, reject) {
          // Simular una operación asíncrona (por ejemplo, una solicitud a un servidor)
  setTimeout(function() {
    // Simulación de éxito
    var exito = true;

    if (exito) {
      // Resuelve la promesa con un valor
      resolve("¡La operación fue exitosa!");
    } else {
      // Rechaza la promesa con un error
      reject("¡Hubo un error en la operación!");
    }
  }, 2000); // Simular una demora de 2 segundos
 });

 // Manejar el resultado de la promesa
 miPromesa.then(function(resultado) {
  // Se ejecuta si la promesa se resuelve exitosamente
  console.log(resultado);
 }).catch(function(error) {
  // Se ejecuta si la promesa es rechazada
  console.error(error);
 });
```

8. **Pregunta de Código:**

- Escribe una función en JavaScript que haga una petición AJAX (usando fetch) para obtener datos de una API y los muestre en un elemento con el id #result.

```
function obtenerDatosDeAPI() {
  // URL de la API
```

```
var url = 'https://api.escuelajs.co/api/v1/products';

// Hacer la petición utilizando fetch
fetch(url)
  .then(function(response) {
    // Verificar si la respuesta es exitosa
    if (!response.ok) {
      throw new Error('Error en la petición: ' + response.status);
    }
    // Convertir la respuesta a formato JSON
    return response.json();
  })
  .then(function(data) {
    // Mostrar los datos en el elemento con el id "result"
    var resultadoElemento = document.getElementById('result');
    resultadoElemento.textContent = JSON.stringify(data);
  })
  .catch(function(error) {
    // Manejar errores de la petición
    console.error('Error al obtener los datos:', error);
  });
}
```

9. **Pregunta Teórica:**

- ¿Cuál es la diferencia entre null, undefined y NaN en JavaScript?

En JavaScript, null, undefined y NaN son valores que indican diferentes tipos de ausencia de valor o errores, null se utiliza para indicar la ausencia intencional de un valor, undefined indica la ausencia de inicialización o de propiedad, y NaN indica un valor que no es un número válido en operaciones aritméticas.

10. **Pregunta de Código:**

- Implementa una función en JavaScript que use localStorage para guardar una clave-valor y luego recuperarla.

```
// Función para guardar un valor en localStorage
function guardarEnLocalStorage(clave, valor) {
  // Convertir el valor a cadena si no es una cadena
  if (typeof valor !== 'string') {
    valor = JSON.stringify(valor);
  }
  // Guardar la clave-valor en localStorage
  localStorage.setItem(clave, valor);
```

```
}

// Función para recuperar un valor de localStorage
function recuperarDeLocalStorage(clave) {
  // Recuperar el valor de localStorage
  var valor = localStorage.getItem(clave);
  // Intentar convertir el valor a su tipo original si es un objeto o array
  try {
    valor = JSON.parse(valor);
  } catch (error) {
    // No hacer nada si no se puede parsear el valor
  }
  return valor;
}
```

Sección 3: Git

1. **Pregunta Teórica:**

- ¿Qué es Git y para qué se utiliza en el desarrollo de software?

Git es un sistema de control de versiones distribuido, es una herramienta que permite a los desarrolladores trabajar en un proyecto de software de manera colaborativa, manteniendo un historial completo de todos los cambios realizados en el código fuente.

2. **Pregunta de Comandos:**

- ¿Cuál es el comando para clonar un repositorio de Git?

Por ejemplo, para clonar un repositorio llamado "ejemplo-repositorio" desde GitHub:

git clone https://github.com/usuario/ejemplo-repositorio

3. **Pregunta Teórica:**

- Explica qué es un "branch" (rama) en Git y para qué se utiliza.

Básicamente, es una versión paralela del código principal que permite a los desarrolladores trabajar en nuevas características, correcciones de errores o experimentos sin afectar el código principal del proyecto.

4. **Pregunta de Comandos:**

- Proporciona los comandos necesarios para crear una nueva rama llamada feature-xyz, cambiar a esa rama, y luego fusionarla con la rama main.

- Este comando crea una nueva rama llamada "feature-xyz" y cambia a esa rama.
  > git checkout -b feature-xyz
- Este comando te lleva a la rama principal del proyecto
  > git checkout main
- Actualizar la rama principal (main) con los últimos cambios remotos
  > git pull origin main
- Fusionar la rama "feature-xyz" con la rama principal (main)
  > git merge feature-xyz
- Si desea eliminar la rama "feature-xyz" una vez que se realizó el merge
  >git branch -d feature-xyz

5. **Pregunta Teórica:**

- ¿Qué es un "merge conflict" y cómo se resuelve?

Un "merge conflict" (conflicto de fusión) que ocurre cuando Git no puede realizar automáticamente la fusión de los cambios entre dos ramas debido a que los cambios en esas ramas afectan las mismas partes del código.

Cuando Git detecta un conflicto de fusión, marca las secciones del código que presentan conflictos y solicita al usuario que resuelva manualmente el conflicto antes de completar la fusión, el usuario debe elegir mantener solo una versión de las líneas en conflicto, combinar las versiones o escribir completamente nuevas líneas que resuelvan el conflicto de manera apropiada. Una vez que hayas resuelto los conflictos, elimina las marcas de conflicto (<<<<<<<, ======= y >>>>>>>) y guarda los cambios en los archivos.

6. **Pregunta de Comandos:**

- ¿Cuál es el comando para visualizar el estado actual del repositorio en Git?

Comando para visualizar el estado actual del repositorio en Git.
 > git status

7. **Pregunta Teórica:**

- Explica la diferencia entre git pull y git fetch.

La diferencia principal entre git fetch y git pull radica en que git fetch solo descarga los cambios del repositorio remoto a tu repositorio local sin fusionarlos automáticamente, mientras que git pull descarga los cambios y los fusiona automáticamente en tu rama local.

8. **Pregunta de Comandos:**

- ¿Cuál es el comando para revertir el último commit en Git?
El comando para revertir el último commit en Git es
 > git reset

9. **Pregunta Teórica:**

- ¿Qué es un "remote repository" y cómo se configura en Git?

Un "remote repository" (repositorio remoto) en Git es una versión del repositorio de código de un proyecto que se encuentra en un servidor remoto, como GitHub, GitLab, Bitbucket, o en otro servidor Git en tu red local.

Configurar un repositorio remoto en Git implica vincular tu repositorio local con un repositorio remoto específico en un servidor remoto. Esto se hace utilizando el comando “git remote”.

Utiliza el comando git remote add para agregar el repositorio remoto a tu repositorio local. Por ejemplo:

git remote add origin <URL_DEL_REPOSITORIO>

10. **Pregunta de Comandos:**

- Proporciona los comandos para añadir todos los cambios en los archivos al staging area y luego realizar un commit con el mensaje "Initial commit".

Añadir todos los cambios al área de preparación (staging area), este comando añade todos los cambios realizados en los archivos al área de preparación. El punto “.” indica que se añadirán todos los archivos modificados y nuevos en el directorio de trabajo
> git add .

Este comando crea un nuevo commit con los cambios añadidos al área de preparación. El mensaje -m se utiliza para proporcionar un mensaje descriptivo al commit, en este caso, "Initial commit"
> git commit -m "Initial commit"

Sección 4: MySQL

1. **Pregunta de Código:**

- Escribe una consulta SQL para crear una base de datos llamada company y una tabla llamada employees con las siguientes columnas: id (INT, auto-increment, primary key), name (VARCHAR(100)), position (VARCHAR(50)), salary (DECIMAL(10, 2)), y hire_date (DATE).

```
CREATE DATABASE IF NOT EXISTS company;

USE company;

CREATE TABLE IF NOT EXISTS employees (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   position VARCHAR(50),
   salary DECIMAL(10, 2),
   hire_date DATE
);
```

2. **Pregunta Teórica:**

- Explica la diferencia entre una clave primaria (Primary Key) y una clave foránea (Foreign Key) en MySQL. ¿Cuándo y por qué se utilizan?

La clave primaria se utiliza para identificar de manera única cada fila en una tabla, una clave primaria es un campo o conjunto de campos que identifica de manera única cada fila en una tabla, mientras que la clave foránea es un campo o conjunto de campos en una tabla que hace referencia a la clave primaria de otra tabla, la clave foránea se utiliza para establecer relaciones entre tablas y garantizar la integridad referencial de los datos

3. **Pregunta de Código:**

- Escribe una consulta SQL para insertar tres registros en la tabla employees creada en la pregunta 2.

Insertar tres registros en la tabla "employees"

```
INSERT INTO employees (name, position, salary, hire_date) VALUES
    ('John Doe', 'Developer', 50000.00, '2024-01-15'),
    ('Jane Smith', 'Manager', 70000.00, '2023-09-20'),
    ('Michael Johnson', 'Sales Representative', 60000.00, '2024-03-10');
```

4. **Pregunta de Código:**

- Muestra cómo actualizar el salario de un empleado específico en la tabla employees. Por ejemplo, actualiza el salario del empleado con id = 1 a 60000.00.

```
UPDATE employees set salary=60000.0 WHERE id=1;
```

5. **Pregunta de Código:**

- Escribe una consulta SQL para seleccionar todos los empleados cuyo salario sea mayor a 50000.00 y ordenarlos por el campo hire_date en orden descendente.

```
SELECT * from employees WHERE salary > 50000;
```

6. **Pregunta Teórica:**

- ¿Qué es una transacción en MySQL y cómo se utiliza? Proporciona un ejemplo de uso.

En MySQL, una transacción es una secuencia de operaciones que se ejecutan como una única unidad lógica de trabajo. Las transacciones en MySQL se utilizan principalmente para garantizar la integridad de los datos y la consistencia de la base de datos en entornos multiusuario, donde múltiples usuarios pueden realizar operaciones concurrentes en la base de datos.

```
-- Iniciar la transacción
START TRANSACTION;

-- Reducir el saldo de la cuenta de origen
UPDATE accounts SET balance = balance - 100 WHERE account_id = 123;

-- Aumentar el saldo de la cuenta de destino
```

```
UPDATE accounts SET balance = balance + 100 WHERE account_id = 456;

-- Confirmar la transacción
COMMIT;
```

En este ejemplo, las operaciones de actualización del saldo de las cuentas de origen y destino se realizan dentro de una transacción. Si ambas operaciones se completan con éxito, la transacción se confirma y los cambios se vuelven permanentes en la base de datos. Si ocurre un error en alguna de las operaciones, se revierten todas las operaciones de la y la base de datos vuelve al estado anterior a la transacción. Esto garantiza que la integridad de los datos se mantenga incluso en caso de errores.

7. **Pregunta de Código:**

- Crea una vista en MySQL llamada high_earning_employees que seleccione todas las columnas de los empleados cuyo salario sea mayor a 70000.00.

```
CREATE VIEW high_earning_employees

AS

SELECT * FROM employees WHERE salary > 70000.00;
```