**Título del examen: Desafío de Integración Multilenguaje y Base de Datos**

**Objetivo:**

Construir una solución distribuida que gestione usuarios y sus transacciones financieras usando PHP, C# y Node.js, interactuando con MySQL y SQL Server.

Requisitos:

* El postulante debe tener instalado en su laptop Visual code, Visual Studio, php y nodejs. Asi como MySQL y SQL Server.
* El postulante deberá tener una cuenta en GitLab pues subirá su solución al final y compartirá su desarrollo.

**Instrucciones generales:**

1. Crea una base de datos en **MySQL** para manejar información de usuarios.
2. Crea una base de datos en **SQL Server** para manejar información de transacciones financieras.
3. Diseña APIs en **PHP**, **C#**, y **Node.js** para realizar las operaciones solicitadas.
4. Asegúrate de documentar el proceso de configuración y ejecución de cada componente.
5. Se permite el uso de frameworks o bibliotecas.

**Requerimientos:**

**Parte 1: Base de datos**

1. **MySQL - Usuarios:**
   * Tabla: users
     + id (int, PK, auto\_increment)
     + name (varchar, 255)
     + email (varchar, 255, único)
     + created\_at (timestamp, default CURRENT\_TIMESTAMP)
2. **SQL Server - Transacciones:**
   * Tabla: transactions
     + id (int, PK, auto\_increment)
     + user\_id (int, FK a MySQL.users.id)
     + amount (decimal(10, 2))
     + transaction\_date (datetime, default CURRENT\_TIMESTAMP)

**Parte 2: Desarrollo de APIs**

Crea una API RESTful para cada lenguaje que cumpla las siguientes funciones:

1. **PHP (Usuarios):**
   * **POST** /users: Agrega un nuevo usuario a la tabla users de MySQL.
   * **GET** /users: Lista todos los usuarios.
   * **GET** /users/{id}: Obtiene los detalles de un usuario específico.
   * **DELETE** /users/{id}: Elimina un usuario y sus transacciones asociadas.
2. **Node.js (Transacciones):**
   * **POST** /transactions: Agrega una nueva transacción a la tabla transactions en SQL Server.
   * **GET** /transactions: Lista todas las transacciones.
   * **GET** /transactions/user/{userId}: Obtiene todas las transacciones asociadas a un usuario.
3. **C# (Reporte):**
   * **GET** /report: Retorna un informe combinado desde MySQL y SQL Server con:
     + Nombre del usuario.
     + Email del usuario.
     + Total de transacciones realizadas.
     + Monto total de las transacciones.

**Parte 3: Comunicación entre APIs**

1. El API de **C#** debe consumir los endpoints de PHP y Node.js para generar el reporte combinado.
2. Implementa manejo de errores para conexiones fallidas o datos inconsistentes.

**Criterios de evaluación:**

1. **Base de datos:**
   * Correcta estructura y relaciones entre las tablas en MySQL y SQL Server.
2. **API RESTful:**
   * Cumplimiento de los requerimientos funcionales.
   * Correcto manejo de respuestas y códigos HTTP.
3. **Integración:**
   * Comunicación efectiva entre los tres lenguajes mediante APIs.
   * Reporte funcional y consistente.
4. **Documentación:**
   * Instrucciones claras para ejecutar el proyecto.
   * Explicación del diseño y las decisiones tomadas.