**Backend Developer Assessment**

Each exercise should be designed to be solved within 30-45 minutes, allowing for a total of 3 hours for completion.

**Exercise 1**: Microservices Architecture

Topic: Java for Microservices

Task:

Explain the key principles and advantages of the microservices architecture. Additionally, outline the challenges and trade-offs associated with adopting a microservices approach.

Response:

Responsabilidad única:

Cada microservicios tiene una funcionalidad específica.

Acoplamiento

Dependencia entre servicios disminuye ya que al implementar casa uno una lógica propia para un objetivo único, no afecta a los demás por los cambios en el microservicio específico.

Reusabilidad: Cada microservicio puede ser utilizado en diferentes procesos del negocio.

Flexibilidad: Al tener su propio ambiente es posible realizar escalamiento de manera independiente basado en la transacionalidad y demanda. Permite implementar diferentes tecnologías que se adapten al menos objetivo del microservicio.

Velocidad en el desarrollo: El desarrollo se vuelve más flexible al tener objetivos concretos.

Despliegue independiente: Cada microservicio puede ser desplegado, escalado y desarrollado de manera independiente permitiendo ciclos de desarrollo más ágiles

Tolerancia a fallos: Si un componente falla, la tolerancia a fallas es la capacidad que permite que un sistema continúe funcionando correctamente. Las fallas de un Microservicio no necesariamente afectan a otros.

**Pros of Microservices**

* Fácil e independiente escalamiento.
* Reusabilidad.
* Facil de entender el código.
* Flexibilidad de tecnología.
* Despliegues independientes.
* Tolerancia a fallos

**Cons of Microservices**

* Relatively Complex Deployment
* Complejidad en gestión.
* Aumento de latencia de red.
* Complejidad de consistencia de datos.
* Sobrecarga en despliegues CI/CD
* Testeo global más complejo.

**Exercise 2**: Enterprise Java

Topic: Enterprise Java

Task:

Implement a Java interface called `EmployeeService` that includes the following methods:

1. `List<Employee> getAllEmployees()`: Retrieves a list of all employees.

2. `Employee getEmployeeById(long id)`: Retrieves an employee by their ID.

3. `void addEmployee(Employee employee)`: Adds a new employee to the system.

**Exercise 3**: Spring Boot REST API

Topic: Spring Boot

Task:

Create a Spring Boot application that exposes a RESTful API for managing books in a library. The API should have endpoints to:

1. Get a list of all books in the library.

2. Get a book by its ISBN number.

3. Add a new book to the library.

**Exercise 4**: Spring Data JPA

Topic: Spring Data JPA

Task:

Create a Spring Data JPA repository for the `Book` entity from the previous exercise. The repository should include methods to:

1. Find books by the author's name.

2. Find books published after a certain date.

**Exercise 5**: Spring Boot and Database Transaction

Topic: Spring Boot and Database Transaction

Task:

In the book library application, modify the `addBook` method in the `BookService` to ensure that the book insertion is performed within a database transaction. The transaction should be rolled back if the book's publication date is in the future (i.e., not published yet).