Evaluación Contínua UF2405: Modelo de programación web y bases de datos

Esta evaluación continua tiene como objetivo aplicar los conocimientos de desarrollo en Java, manejo de bases de datos SQL y uso de Spring Boot para construir aplicaciones que interactúen de manera eficiente con una base de datos. La evaluación consta de dos ejercicios principales, en los que se espera que se sigan las mejores prácticas posibles en el desarrollo de cada componente.

Ejercicio 1: Aplicación Java Maven con DAO (5 puntos)

Objetivo: Implementar una aplicación Java utilizando Maven que se conecte con una base de datos relacional (por ejemplo, MySQL) y que implemente el patrón DAO para realizar operaciones CRUD. Este ejercicio tiene como fin consolidar conocimientos sobre programación orientada a objetos, el uso de DAOs para la comunicación con la base de datos y la configuración de un proyecto Maven en Java.

1. **Configuración del Proyecto (Maven):**
    - Configurar un proyecto Maven, asegurando la correcta configuración del archivo `pom.xml` con las dependencias necesarias, como el conector JDBC para la base de datos.
2. **Clases Base:**
    - Crear dos clases base que representen entidades de datos, como `Producto` y `Cliente`.
    - Cada clase debe incluir atributos básicos y relevantes (por ejemplo, `id`, `nombre`, `precio` para `Producto`, y `id`, `nombre`, `correo` para `Cliente`), así como métodos `getter` y `setter`, un constructor y un método `toString()`.
3. **Clases DAO:**
    - Implementar clases DAO (`ProductoDAO` y `ClienteDAO`) que permitan la gestión de operaciones CRUD para cada clase.
    - Definir métodos básicos como `crear`, `actualizar`, `borrar` y `consultar`.

4. **Conexión a la Base de Datos:**
    - Crear una clase `DatabaseConnexion` para gestionar la conexión con la base de datos, aplicando el patrón Singleton para garantizar una única instancia de la conexión.
5. **Clase Main:**
    - Desarrollar una clase `Main` en la que se realicen demostraciones de las operaciones CRUD mediante las clases DAO, incluyendo ejemplos de inserción, actualización, eliminación y consulta.
6. **Base de Datos:**
    - Definir y crear una base de datos SQL con tablas correspondientes a `Producto` y `Cliente`, y proveer un script SQL para su creación.

Ejercicio 2: Aplicación Spring Boot con Repositorios y Controladores (5 puntos)

Objetivo: Migrar la aplicación a un entorno Spring Boot, reutilizando las clases base (`Producto` y `Cliente`) y la base de datos del primer ejercicio. La aplicación debe exponer un servicio RESTful que permita la interacción desde un cliente web básico. Este ejercicio se centra en el desarrollo de aplicaciones basadas en servicios, creando una interfaz que consuma los endpoints REST de la aplicación.

1. **Configuración del Proyecto (Spring Boot):**
    - Configurar un proyecto Spring Boot, asegurando que el archivo `pom.xml` incluya las dependencias necesarias (Spring Data JPA, Spring Web, conector JDBC, entre otras).
2. **Clases Base y Repositorios:**
    - Reutilizar las clases `Producto` y `Cliente` del primer ejercicio.
    - Implementar interfaces de repositorio `ProductoRepository` y `ClienteRepository` extendiendo `JpaRepository` para gestionar operaciones CRUD.
3. **Controladores:**
    - Crear controladores (`ProductoController` y `ClienteController`) para exponer las operaciones CRUD mediante endpoints REST.

- Definir rutas como `GET /productos`, `POST /productos`, etc., para cada entidad.

4. **Configuración del Proyecto:**
   - Implementar una clase `WebConfig` para configurar aspectos básicos del proyecto.
   - Configurar `application.properties` con las propiedades de conexión a la base de datos.
5. **Interfaz de Usuario (Frontend):**
   - Crear un cliente sencillo con un archivo HTML, uno de JavaScript y otro de CSS.
   - **HTML**: Diseñar una interfaz básica para visualizar y manipular productos y clientes.
   - **JavaScript**: Implementar interacciones con los endpoints REST de la aplicación.
   - **CSS**: Aplicar estilos básicos a la interfaz.
6. **Base de Datos:**
   - Reutilizar la misma base de datos y tablas del primer ejercicio, asegurando que la configuración de conexión en `application.properties` sea correcta.

Reutilización y Buenas Prácticas

El ejercicio está diseñado para que se reutilicen las clases base (`Producto` y `Cliente`) y la misma base de datos en ambos ejercicios. Esto permite mantener consistencia en la lógica de negocio y los datos. Se espera que se sigan siempre las mejores prácticas de desarrollo en cuanto a organización del código, modularidad y documentación de los componentes.

Entrega de la Evaluación Continua

Para completar la entrega de esta evaluación continua, se debe preparar lo siguiente:

1. **Proyectos en formato ZIP**:
    - Cada uno de los dos ejercicios debe entregarse en un archivo ZIP separado.
    - Ambos archivos ZIP, junto con el archivo de la base de datos, deben estar contenidos dentro de una carpeta comprimida adicional.
2. **Opciones de entrega**:
    - Subir la carpeta comprimida a una plataforma en la nube, como Google Drive, GitHub u otro aplicativo similar.
    - Compartir el enlace en el espacio dedicado a ello en la plataforma del campus de Ironhack.
3. **Permisos de acceso**:
    - Asegurarse de otorgar acceso al proyecto al correo electrónico sergi.faura@ironhack.com para facilitar la revisión.

Rúbrica de Evaluación

Rúbrica de Evaluación para Ejercicio 1 (5 puntos)

1. **Configuración del Proyecto Maven y Dependencias (1 punto)**
    - **1 punto**: El proyecto Maven está correctamente configurado, el archivo `pom.xml` incluye todas las dependencias necesarias, se sigue la estructura de carpetas adecuada y no presenta errores de compilación.
    - **0.5 puntos**: El proyecto está configurado, pero faltan algunas dependencias o se presentan advertencias en la compilación.
    - **0 puntos**: El proyecto no está configurado correctamente o presenta errores de compilación graves.
2. **Clases Base (Producto y Cliente) (1 punto)**
    - **1 punto**: Las clases base cumplen con los requisitos, tienen los atributos especificados, métodos `getter` y `setter`, constructor y el método `toString()`.
    - **0.5 puntos**: Las clases base están presentes, pero faltan algunos métodos o atributos especificados.
    - **0 puntos**: Las clases base no cumplen con los requisitos o presentan errores importantes.
3. **Clases DAO (ProductoDAO y ClienteDAO) (1.5 puntos)**
    - **1.5 puntos**: Las clases DAO están correctamente implementadas y cubren las operaciones CRUD (crear, actualizar, borrar, consultar) para cada entidad.
    - **1 punto**: Las clases DAO están implementadas pero alguna de las operaciones CRUD es incompleta o presenta errores.
    - **0.5 puntos**: Las clases DAO tienen errores significativos o faltan varias operaciones CRUD.
    - **0 puntos**: Las clases DAO no están implementadas correctamente.
4. **Conexión a la Base de Datos (DatabaseConnexion) (0.5 puntos)**
    - **0.5 puntos**: La clase `DatabaseConnexion` está implementada correctamente con el patrón Singleton y establece la conexión con la base de datos sin errores.
    - **0 puntos**: La clase de conexión presenta errores o no sigue el patrón Singleton.

5. **Clase Main y Demostración CRUD (1 punto)**
    - **1 punto**: La clase `Main` incluye demostraciones funcionales de las operaciones CRUD para ambas entidades.
    - **0.5 puntos**: La clase `Main` incluye demostraciones CRUD, pero no todas funcionan correctamente.
    - **0 puntos**: La clase `Main` no contiene demostraciones funcionales de las operaciones CRUD.

Rúbrica de Evaluación para Ejercicio 2 (5 puntos)

1. **Configuración del Proyecto Spring Boot y Dependencias (0.5 puntos)**
    - **0.5 puntos**: El proyecto Spring Boot está configurado correctamente con las dependencias necesarias en `pom.xml`, se sigue la estructura de carpetas adecuada y se ejecuta sin problemas.
    - **0 puntos**: El proyecto no está configurado correctamente o presenta errores de ejecución.
2. **Repositorios (ProductoRepository y ClienteRepository) (1 punto)**
    - **1 punto**: Los repositorios están implementados correctamente, extendiendo `JpaRepository` y permiten realizar operaciones CRUD sobre las entidades.
    - **0.5 puntos**: Los repositorios están presentes pero tienen errores menores o no cumplen completamente las operaciones CRUD.
    - **0 puntos**: Los repositorios no están implementados o presentan errores importantes.
3. **Controladores (ProductoController y ClienteController) (1.5 puntos)**
    - **1.5 puntos**: Los controladores están implementados correctamente, exponen endpoints REST y permiten realizar todas las operaciones CRUD.
    - **1 punto**: Los controladores están presentes pero tienen errores menores o alguna operación CRUD no funciona correctamente.
    - **0.5 puntos**: Los controladores tienen errores significativos o faltan varias operaciones CRUD.
    - **0 puntos**: Los controladores no están implementados correctamente.

4. **Configuración y Archivo application.properties (0.5 puntos)**
    - **0.5 puntos**: La configuración de `WebConfig` y `application.properties` está correctamente implementada, permitiendo la conexión y ejecución de la aplicación sin errores.
    - **0 puntos**: La configuración de conexión y `application.properties` presenta errores o no permite la conexión con la base de datos.
5. **Interfaz de Usuario (HTML, JavaScript y CSS) (1.5 puntos)**
    - **1.5 puntos**: La interfaz de usuario está correctamente implementada con un archivo HTML, uno de JavaScript y otro de CSS, y permite visualizar e interactuar con los endpoints REST.
    - **1 punto**: La interfaz de usuario está implementada pero tiene algunos errores menores en la interacción con los endpoints REST.
    - **0.5 puntos**: La interfaz de usuario presenta errores significativos o no permite la interacción completa con los endpoints.
    - **0 puntos**: La interfaz de usuario no está implementada o no permite la interacción con los endpoints REST.

Recordatorio: Buenas Prácticas de Código y Estructura de Carpetas

Para ambos ejercicios, es fundamental seguir buenas prácticas de desarrollo. Esto incluye:

- Nombres de variables claros y representativos.
- Uso adecuado de las convenciones de Java.
- Estructura de carpetas organizada y adecuada para cada ejercicio.
- Documentación breve y relevante en el código (comentarios).

Puntuación máxima: **10 puntos**