AccesData SL

Se nos ha contratado para realizar el sistema de seguimiento de Proyectos informáticos de nuestra empresa DataAccessSL. Para ello partimos que vamos a usar una **Base de Datos Relacional y que el despliegue de esta será usando Docker**.

Nuestra empresa está formada por distintos **Departamentos de Desarrollo** los cuales tienen asignados un nombre, jefe de departamento, un presupuesto y una lista de proyectos terminados, así como proyectos en desarrollo, unidos a un presupuesto anual. El jefe de departamento es un programador, que no participa activamente en ningún proyecto en desarrollo, y que puede variar a lo largo del tiempo. Es interesante tener un histórico de jefes que ha tenido un departamento.

Los **Proyectos** tienen un jefe de proyecto, un programador que no puede ser director del departamento ni programador activo del mismo, un nombre, un presupuesto, una fecha de inicio y fin, una lista de tecnologías usadas y un repositorio donde se almacena.

Sobre los **Programadores**, nos interesan saber su nombre, su fecha de alta, así como el departamento al que pertenece, los proyectos activos donde participa, y los commits e issues que realiza para analizar su productividad y las tecnologías que domina y el salario por el que ha sido contratado. Además un programador tiene una contraseña de acceso en hash SHA-256, pero nunca se muestra en peticiones de lectura.

De los **Repositorios**, nos interesa su nombre, su fecha de creación, proyecto asociado y la lista de commits asociadas y los issues existentes.

De los **Commits**, nos interesa titulo, texto, fecha, repositorio y proyecto asociado y autor de este commit e issue de donde viene asignado. No podrá realizar ningún commit ningún programador que no esté en este proyecto ni que no nazca de un issue. Hacer un commit implica cerrar una issue

De las **Issues**, nos interesa, titulo, texto, fecha, y lista de programadores asignados para resolverlo, por supuesto el proyecto y repositorio asignado. Las issue las crea solamente el jefe de Proyecto. No se podrá asignar ningún programador que no esté en proyecto. Además, pueden tener un estado: pendiente o terminada.

Se nos pide resolver este problema y mostrar su funcionamiento teniendo en cuenta que hay que hacer dos proyectos:

1. Realizar el Diagrama de Clases asociado, mostrando la semántica, navegabilidad y cardinalidad de las relaciones, justificando la respuesta. Crear las Clases del modelo asociadas así como las tablas para el almacenamiento en una base de datos relacional.

1 punto

2. Implementación del sistema y realización del Mapeo Objeto Relacional de forma manual para implementar las operaciones CRUD de las entidades relevantes aplicando las restricciones indicadas. Se valorará el conjunto de técnicas, patrones y elementos usados para la implementación de dicho proyecto y justificación de este, respetando la bidireccionalidad siempre que sea posible.

2,5 puntos

3. Implementación del sistema y realización del Mapeo Objeto Relacional usando JPA con Hibernate para implementar las operaciones CRUD de las entidades relevantes aplicando las restricciones indicadas. Se valorará el conjunto de técnicas, patrones y elementos usados para la implementación de dicho proyecto y justificación de este, respetando la bidireccionalidad siempre que sea posible.

2,5 puntos

4. Se debe probar en los dos proyectos dichas operaciones CRUD usando JUnit5 o JUnit4. Intenta simular la base de datos con datos de prueba.

1,5 punto

5. Además, de las operaciones CRUD de obligatorio cumplimento con las restricciones semántica indicadas y **solo haciendo uso de ellas lo máximo posible, y minimizando el número de llamadas al servidor** nos interesa saber además las siguientes:

1) Obtener de un departamento, los proyectos (información completa) y trabajadores asociados con sus datos completos.
2) Listado de issues abiertas por proyecto que no se hayan consolidado en commits.
3) Programadores de un proyecto ordenados por número de commits.
4) Programadores con su productividad completa: proyectos en los que participa, commits (información completa) e issues asignadas (información completa).
5) Obtener los tres proyectos más caros en base a su presupuesto asignado y el salario de cada trabajador que participa.

1,5 punto

6.- Debemos tener en cuenta que necesitamos devolver cualquier salida pedidas en los puntos anteriores en JSON y XML, según sea llamado, usando un parámetro para diferenciar cuando queremos una u otra. No se acepta otro formato, ni otro tipo de salida.

1 punto

Nuestro programa debe llamarse con un JAR de la siguiente manera

java -jar accesdata.jar. Se recomienda que el proyecto esté gestionado por Maven.

Se debe entregar:

- **Repositorio GitHub Personal y el de entrega** con la solución en el que incluyas:

- Readme explicando el proyecto y el nombre de los integrantes. Usa Markdown y mejora su estilo. Si no perderás puntos por la presentación.

- **Código fuente comentado y perfectamente estructurado de los dos proyectos** con JDoc. Además de los gitignore adecuados y que siga el flujo de trabajo GitFlow.
- No se deben incluir los ejecutables si no se deben poder crear los **jar.**
- **Docker con la Base de Datos que incluya datos de ejemplo para probar la aplicación desde su inicio o fichero de SQLite.**

- **Documentación en PDF** donde expliques el diseño y propuesta de solución, así como clases y elementos usados haciendo especial énfasis en:

  Diagrama de clases y justificación de este.
  Arquitectura del sistema y patrones usados.
  Explicación de forma de acceso a los datos.
  La no entrega de este fichero invalidará la práctica.
  La aplicación no debe fallar y debe reaccionar antes posibles fallos asegurando la consistencia y calidad de esta.

- **Enlace en el readme al vídeo en YouTub**e donde se explique las partes más relevantes de la practica y **se muestre su ejecución**. La duración del vídeo debe ser unos 20 minutos. **La no entrega de este vídeo y donde se vea su ejecución anulará el resultado de la práctica**.

- Repositorio oficial de la entrega Enlace de entrega: https://classroom.github.com/a/m1KcCQgq. La subirán los dos miembros del equipo, si no está en este repositorio se invalidará la práctica no pudiéndose entregar por otros medios.

NOTA:

La práctica no es obligatoria, pero no realizarla implica que los Resultados de Aprendizaje no se podrán evaluar a través de ella, lo que implica que el porcentaje de calificación asociado a este instrumento quedará calificado con un NO APTO y con ello el Resultado de Aprendizaje estará calificado como NO APTO y por lo tanto deberán ser evaluados por otros instrumentos como un examen práctico en el periodo de recuperación, de acuerdo con lo establecido en la Programación Didáctica.

Aprobar la práctica no implica que no se haga el examen teórico-práctico asociado al Resultado de Aprendizaje, pues para considerarse superado hay que aprobar los dos instrumentos (práctica y examen) y que la media de ambos sea mayor o igual a 5.

La copia de la práctica o fragmentos de ella implica la evaluación de todos Resultados de Aprendizaje con un 0, no pudiéndose recuperar hasta la evaluación ordinaria.

Fecha de entrega 3 de diciembre de 2021 a las 09:00.