TENNIS LAB BARCELONA

En esta práctica vamos a continuar con la anterior, pero analizando posibles mejoras en su implementación aplicando un diseño y desarrollo NoSQL partiendo de las restricciones del enunciado anterior.
Debemos tener en cuenta que ahora os propios id internos (no los uuid) serán los propios generados por el servicio que se está utilizando.

En nuestra aplicación se conectan distintos **usuarios.** Pero debemos tener en cuenta que este servicio lo tenemos externalizado, por lo que debemos hacer uso de una **API REST** para operar con **todo** lo relacionado con él desde el endpoint: **https://jsonplaceholder.typicode.com/users, estando este servicio cacheado y sincronizado en nuestro sistema para las operaciones CRUD y que que la cache se refresca cada 60 segundos de manera automática.**

Trabajamos con varias **máquinas**, que son de **encordar** o de **personalización, como ya sabemos**.

Los **pedidos pueden estar formados por varias tareas o partes de trabajo a realizar** que recibimos de un **tenista** y son asignados a un **encordador** y tienen un **estado.** Tenemos un **tope de entrega** marcado por una fecha. Los **pedidos** tiene una fecha de entrada, una fecha de salida programada y de salida final y un precio asociado que es la suma de todas las acciones. La fecha de salida final será inicialmente la programada, luego se actualizará a la real tal y como vimos en la práctica anterior. Debemos tener en cuenta restricciones

Para cada **tarea del pedido o acción a realizar a parte de almacenarla en nuestra base de datos la vamos a almacenar remotamente en un histórico en la nube al que podremos acceder mediante el endpoint: https://jsonplaceholder.typicode.com/todos.** Necesitamos saber la **raqueta** o raquetas (una acción por raqueta) con la que trabajar si se necesita. Teniendo en cuenta todas las tareas posibles, precios y restricciones del la práctica anterior.

Debemos tener en cuenta que un **encordador** no puede tener más de dos **pedidos** activos por turno. Del **turno** nos interesa saber comienzo y fin del mismo. Un **encordador** no puede usar otra máquina si ya tiene asignada una en un turno determinado.

Además, como vendemos distintos **productos del tipo**: raquetas, cordajes, y complementos como overgrips, grips, antivibradores, fundas, etc. Necesitamos saber el tipo, marca, modelo, precio y stock del mismo. Ten en cuenta que **solo podrá realizar operaciones CRUD el encargado del sistema**, pero la asignación de pedidos la pueden hacer también los encordadores.

Por otro lado, nos interesa mantener el **histórico de los elementos del sistema** y
- Sistema de gestión de usuarios local y remoto sincronizados.
- CRUD completo de los elementos que consideres necesarios.
- Sistema de errores y excepciones personalizados.
- Información completa en JSON de un pedido.
- Listado de pedidos pendientes en JSON.
- Listado de pedidos completados en JSON.
- Listado de productos y servicios que ofrecemos en JSON.
- Listado de asignaciones para los encordadores por fecha en JSON.
- Gestión de tareas locales y remoto, histórico remoto.

1. Completar la información que te falta hasta tener los requisitos de información completos.

2. Realizar el Diagrama de Clases asociado, mostrando la semántica, navegabilidad y cardinalidad de las relaciones, justificando la respuesta con el máximo detalle. Obtener las colecciones existentes y aplicar patrones de diseño NoSQL razonando en cada punto la decisión adoptada y posibles mejoras en la resolución.

(2 puntos)

3. Implementación y test de repositorios, servicios y controladores **reactivos** de las operaciones CRUD y otras operaciones relevantes aplicando las restricciones indicadas usando solo MongoDB y un framework de inyección de dependencias. Elegir una colección en cada implementación y escuchar sus cambios en tiempo real.

(4 puntos)

4. Implementación y test de repositorios, servicios y controladores **reactivos** de las operaciones CRUD y otras operaciones relevantes aplicando las restricciones indicadas usando Spring Data MongoDB usando el sistema propio de inyección de dependencias que tiene Spring Data. Elegir una colección en cada implementación y escuchar sus cambios en tiempo real.

(4 puntos)

En las dos posibles implementaciones **Se deben tipificar los resultados tanto correctos como incorrectos, como operaciones válidas o inválidas con el sistema que creas más acertado y mostrar as salidas en JSON de manera correcta con los DTOs adecuados.**

Nuestro programa debe llamarse con un JAR de la siguiente manera: **java -jar tennislab.jar**.

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** con JDoc/KDoc. 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 desde el propio proyecto**. **Asegúrate que se puede crear y que los ficheros de configuración de la base de datos, así como datos de ejemplo están en directorios que se pueden ejecutar o se pueden leer desde resources.**
  - **Documentación en PDF** donde expliques el diseño y propuesta de solución, así como clases y elementos usados haciendo especial énfasis en:
    Requisitos de Información.
    Diagrama de clases y justificación de este.
    Alternativas y justificación detallada del modelo NoSQL.
    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 30 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/GSctLb_o. 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 las dos instrumentos (práctica y examen) y que la media de ambos sea mayor o igual a 5 siempre y cuando una de ellas sea mayor que 4.

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 febrero de 2023 a las 23:00.

Información:
https://www.tenniswarehouse-europe.com/lc/RacquetStringTerms.html?lang=es
https://www.tenniswarehouse-europe.com/?lang=es

https://www.tennis-point.es/

“La verdadera amistad debe apoyarse en las cosas que compartimos
y en las que compartiremos en el futuro,
sin ninguna intención de conseguir nada a cambio de ella”
Jorge Bucay