Universidad Francisco de Vitoria UFV Madrid

NOMBRE: _____

GRUPO: _____ FECHA: _____

CALIFICACIÓN:

EXAMEN FINAL – PRÁCTICA (60%)

DESARROLLO E INTEGRACIÓN DEL SOFTWARE

Instrucciones

El razonamiento hasta las soluciones tiene la misma importancia que la propia solución. Se valorará positivamente un razonamiento adecuado. Lee detenidamente todo el ejercicio antes de empezar y con ello poder elegir el orden en que debes contestar cada apartado.

La duración de esta parte es de 140 minutos. Y 10 minutos para la entrega.

EJERCICIO P.1 (10 puntos)

Requisitos y pautas generales de trabajo:

1. Acceder a la siguiente URL de Github Classroom: https://classroom.github.com/a/sitlmvza.
2. Seguir el procedimiento de Github para la creación del repositorio.
3. Clonar el repositorio que ha sido creado para el alumno, o añadir el repositorio como remote a un repositorio local.
4. Trabajar en el enunciado del ejercicio en el repositorio en local.
    a. **Es obligatorio realizar al menos un commit cada 15 minutos.**
5. Una vez dado por concluido el ejercicio, se deberá:
    a. **Realizar un commit final y crear una release v1.0.**
    b. Comprimir la carpeta de trabajo y subirla a la tarea habilitada en Canvas para tal fin.
    c. **Es imprescindible para dar por válida la prueba incluir la carpeta .git**
    d. Eliminar la carpeta target/ para asegurar que el tamaño del fichero comprimido es razonable y no dé problemas con la subida al aula virtual.
6. Debéis entregar también un fichero de texto plano, llamado **referencias.txt**, en el que tendréis que referenciar todas las fuentes que hayáis usado. Ya sean los repositorios de vuestras prácticas o repositorios propios que tengáis con ejercicios. Asimismo, toda referencia externa, por ejemplo, a StackOverflow, debe ser también referenciada en este fichero. Con poner el enlace a la referencia, es suficiente.

Enunciado

Desarrolla una aplicación web que permita gestionar las altas de productos, así como mostrar todos los productos ya almacenados en una tabla. Se debe hacer un backup en pdf de cada producto dado de alta en el sistema.

Mediante el framework Vaadin, realizar una web que contenga:

- Un formulario de alta de producto en la aplicación solicitando al usuario los siguientes datos:
  - Nombre
  - Categoría
  - Precio
  - EAN13.
    - Los números EAN podrán generarse de manera aleatoria en el siguiente enlace y usarlos para la aplicación: https://generate.plus/en/number/ean
- Una tabla o grid que permita visualizar toda la información de los elementos almacenados y que, cada vez que se dé de alta un nuevo elemento, se refresque automáticamente con la información añadida.

Mediante las librerías definidas en el documento se deberá generar un documento **pdf** con la información de cada producto añadido al inventario.

Al añadir el elemento al inventario, es necesario mostrar una notificación en pantalla que indique si el fichero se ha generado correctamente.

La comunicación con el backend se realizará mediante llamadas HTTPRequest a una API RESTful que se encargará de recibir las peticiones con los datos introducidos y llamar al método para generar el fichero pdf con los datos almacenados.

Todos los productos que se den de alta se almacenarán en un fichero JSON que se persistirá en disco y que será usado como base de datos para futuras ejecuciones de la aplicación.

1) Requisitos y pautas del desarrollo front

1. La versión de Java será Amazon Corretto 11.
2. Utilización del framework **Vaadin** 14.9.1
3. Todas las dependencias deberán ser gestionadas por **Maven**.
4. Para la creación del proyecto, se debe trabajar con la plantilla para Vaadin 14 disponible en el siguiente enlace: https://vaadin.com/hello-world-starters.
5. Dependencias adicionales
   a. GroupID: com.google.code.gson
   b. Artifact ID: gson
   c. Version: 2.6.2
6. **Definir el Group ID del proyecto a "es.ufv.dis.final2022" y el Artifact ID con las iniciales del nombre del alumno.**
7. La interfaz deberá contener el formulario de alta de los datos mencionados anteriormente. Se valorará positivamente un diseño de la UI adecuado.
8. **Comunicación con la API**
   a. Las llamadas a la API se realizarán mediante el uso de HTTP Requests, disponibles en Java 11.

2) Requisitos de la API

La API REST recibirá las peticiones desde la aplicación web o frontend. Estará compuesta por dos métodos. Uno POST, encargado de gestionar los datos recibidos en el body de la petición y otro GET que devolverá todos los datos almacenados en el archivo JSON usado como base de datos. Este, que contendrá un listado de datos iniciales a completar sobre el fichero facilitado, usados como ejemplo.

Para generar la API, se hará uso de Spring Initializr. Recordad seleccionar la **versión 2.7.7** de Spring, para garantizar la compatibilidad con vuestra versión de Java. Los controladores de la API se apoyarán en una clase Java extra para almacenar los ficheros pdf.

Todos los documentos pdf generados se guardarán en la carpeta **/productos**, en la raíz del proyecto.

Clase PDFManager

- Se encargará de instanciar y trabajar con la librería.
- Contiene un método de generación del documento pdf.

Dependencias requeridas

Group ID: com.lowagie Artifact ID: itext Version: 4.2.2

Ejemplo de generación de un documento pdf.

```
try {

     Document doc = new Document(PageSize.A4, 50, 50, 100, 72);

     PdfWriter writer = PdfWriter.getInstance(doc, new
FileOutputStream("pageNumbersWatermark.pdf"));

     doc.open();
     String text = "some padding text";
     Paragraph p = new Paragraph(text);
     p.setAlignment(Element.ALIGN_JUSTIFIED);
     doc.add(p);
     doc.close();
 }
     catch ( Exception e ) {
     e.printStackTrace();
 }
```

En este ejemplo solo está la implementación del método de creación del documento básico. Podéis consultar más información en Internet sobre la librería para formatear el documento de la manera adecuada.

3) Requisitos y pautas de las pruebas

Deberán generarse las siguientes pruebas unitarias con la librería indicada.

- Creación de un producto.
- Añadir un producto al inventario.

Dependencia requerida

Group ID: junit Artifact ID: junit Version: 4.12

Entregables

1. La carpeta de ambos proyectos y la carpeta .git del proyecto.
2. Documentos pdf generados por la aplicación en la carpeta **/productos** en la raíz del proyecto backend.
3. Código de los proyectos alojado en el repositorio de Github.
4. Fichero referencias.txt correctamente cumplimentado.

Calificación del ejercicio

A continuación, se indica cuál es la distribución de las notas de este examen:

Apartado	A	B	C
Puntuación	4	4	2

Rúbrica de evaluación

	VALORACIÓN			
DIMENSIÓN	**0 puntos**			**10 puntos**
Implementación de los métodos de la API	No implementa los métodos de la API solicitados en el enunciado	Hay una definición de los métodos solicitados, pero no compilan correctamente o presentan errores de funcionamiento.	Los métodos están definidos y realizan la acción solicitada, pero no recuperan los datos correctamente.	Los métodos están correctamente definidos y recuperan la información solicitada correctamente.
Utiliza las funcionalidades de Git de forma correcta	No existe control de versiones	Control de versiones excesivamente sencillo	Utiliza las funcionalidades justas de forma correcta.	Utiliza todas las funcionalidades estudiadas de forma correcta.
Realiza los tests de la aplicación mediante el uso correcto del Framework JUnit.	Los tests no existen o no están bien definidos y no pueden ser ejecutados correctamente.	Los tests existen, pero no cubren los casos planteados en el enunciado.	Los tests tienen pequeños errores de diseño e implementación.	Los tests son completos y no tienen errores.

Interfaz implementada mediante Vaadin 14.	La interfaz no se ha implementado.	La interfaz existe, pero no dispone de todos los campos solicitados ni es capaz de enviar las peticiones ni mostrar la información.	Existen todos los campos de formulario solicitados, pero existen errores en los tipos o el proceso de envío falla. Los datos se visualizan, pero de forma incorrecta.	La interfaz funciona correctamente, dispone de todos los campos solicitados y permite enviar las solicitudes, así como muestra los datos correctamente.
Uso correcto de la librería HttpRequest de Java 11	No usa la librería HttpRequest de Java 11.	Se hace uso de la librería, pero no corresponde con lo que se solicita en el enunciado. La estructura de los métodos llamados no es correcta.	Se hace uso correcto de la librería, pero existen errores de concepto en la gestión de los métodos o en los datos devueltos.	El uso de los métodos de la librería es el correcto y los datos recuperados son los correctos.
Argumenta correctamente las decisiones de desarrollo y las basa en la experiencia previa.	No basa las decisiones en razonamientos lógicos o relacionados con lo visto en la asignatura.			Las decisiones de diseño e implementación están basadas en razonamientos lógicos desarrollados a partir de los conocimientos adquiridos en la asignatura.
Generación de los datos solicitados, tanto los ficheros como los nuevos objetos	No se generan los ficheros pdf para cada uno de los elementos nuevos.	Los ficheros generados son incorrectos o no coincide el nombre con el especificado. Los objetos no se añaden o se añaden al fichero JSON pero no son correctos.	Los ficheros generados son correctos pero el nombre no coincide. Los objetos se añaden al fichero JSON pero no son correctos.	Todos los ficheros generados son correctos en formato y nombre y los datos se añaden correctamente al fichero JSON.