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/GABMwERI.
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 las carpetas ***target/*** y ***/node_modules*** 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. En este fichero, también es imprescindible añadir todas las conversaciones mantenidas con **ChatGPT**. Por favor, no dejéis de añadir estas referencias para evitar situaciones incómodas con temas de plagio.
7. **IMPORTANTE**: Es obligatorio realizar las peticiones e intercambios de datos mediante **HTTP requests**, utilizando Gson para serializar y deserializar las clases.
   Queda terminantemente prohibido manejar ficheros directamente para este propósito
   El incumplimiento de este requisito supondrá la suspensión automática de la prueba.
8. **IMPORTANTE**: Es obligatorio utilizar un flujo de trabajo con Git siguiendo la estructura de GitFlow. Deberán existir, como mínimo, las ramas principales:

- main o master (rama principal),
- develop (rama de desarrollo),

y se deberán crear ramas con el prefijo adecuado para las funcionalidades (feature/) y lanzamientos (release/).

El incumplimiento de este requisito supondrá la suspensión automática de la prueba.

Enunciado

Disponemos de una base de datos de usuarios en un fichero JSON (usuarios.json), con los usuarios de una aplicación CMS para gestionar los clientes y las ventas de una empresa que no mencionaremos por motivos de Copyright.

Se pide una aplicación web o frontend que muestre por pantalla los datos del fichero JSON mencionado anteriormente. Los datos se recuperarán mediante una llamada a una API REST que se explicará en el siguiente apartado.

Los datos deberán mostrarse en un grid o tabla nada más cargar la aplicación, sin que sea necesario pulsar ningún botón.

Mediante el framework Vaadin, realizar una web que contenga:

- Un grid con las columnas correspondientes a los datos contenidos en el fichero para cada uno de los usuarios. Los nombres de las columnas serán los nombres de las claves del fichero JSON, en un formato correcto (primera letra en mayúsculas, espacios entre palabras, etc.). **No** es necesario incluir en el grid una columna para cada una de las claves del fichero json. Por ejemplo, todos los datos relativos a la dirección y los métodos de pago, no es necesario que se incluyan.
- Al hacer doble click sobre una de las filas del grid, debe abrirse un diálogo modal que contenga la información de la fila completa. Se debe incluir la información relativa a dirección y a métodos de pago en este diálogo modal.
- Debe haber un botón en cada fila del grid que contenga el texto “Editar”. Al pulsar este botón, se mostrará la información relativa al usuario, incluyendo los datos de dirección y método de pago, permitiendo editarlos.
- Debajo del grid debe haber un botón llamado “Añadir usuario”. Este método mostrará un diálogo modal con todos los campos necesarios para crear un nuevo usuario. El id se debe generar automáticamente.
- Debajo del grid debe haber un botón llamado “Generar PDF”. Este método hará una llamada a la API para dar la orden de generar el fichero pdf con los datos contenidos en el fichero json.
- No es necesario eliminar filas del grid.

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. En el siguiente apartado, se define el método en la API para generar el fichero.

1) Requisitos y pautas del desarrollo front

1. La versión de Java será **Amazon Corretto 17**.
2. Utilización del framework **Vaadin 24**.
3. Todas las dependencias deberán ser gestionadas por **Maven**.
4. Para la creación del proyecto, se debe trabajar con la plantilla para Vaadin 24 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: 2.11.0
    d. Es imprescindible usar GSON para la serialización/deserialización de los ficheros JSON en el front.
6. **Definir el Group ID del proyecto a "es.ufv.dis.front.final2025" y el Artifact ID con las iniciales del nombre del alumno.**
7. La interfaz deberá contener el grid indicado en el enunciado, incluyendo el botón "Editar" en la última columna. Se valorará positivamente un diseño de la UI adecuado. A modo de ayuda, os dejamos a continuación una manera de añadir el botón a la última columna:

```
grid.addColumn(
        new NativeButtonRenderer<>("Editar",
                clickedItem -> {
                    // Llamada al método de editar
                })
);
```

8. **Comunicación con la API**
    a. Las llamadas a la API se realizarán mediante el uso de **HTTP Requests**, disponibles a partir de Java 11.

2) Requisitos de la API

La API REST recibirá las peticiones desde la aplicación web o frontend. Estará compuesta por cuatro métodos:

POST: Recibirá en el body de la petición un fichero con la información a añadir al fichero.

GET: Devolverá todos los datos almacenados en el archivo JSON usado como base de datos. Este, contendrá el listado de naves facilitado al principio del examen.

GET/{id}: Devolverá la información asociada al usuario identificado con el id pasado como parámetro.

PUT/{id}: Recibirá en el body de la petición la información a editar al fichero para el identificador pasado como parámetro en la url del endpoint.

Será necesario generar una nueva petición **GET** con un endpoint diferente que permita indicar al backend que se debe generar el fichero pdf.

La gestión de los ficheros JSON en el Back, se realizará mediante el uso de la librería **GSON.**

Para generar la API, se hará uso de Spring Initializr. Recordad seleccionar la **versión 3.4.1** de Spring, para garantizar la compatibilidad con vuestra versión de Java.

Definir el Group ID del proyecto a "es.ufv.dis.back.final2025" y el Artifact ID con las iniciales del nombre del alumno.

Dependencias adicionales

1. GroupID: com.google.code.gson
2. Artifact ID: gson
3. Version: 2.11.0

Es imprescindible usar GSON para la serialización/deserialización de los ficheros JSON en el back.

Gestión de los ficheros pdf.

Los métodos de la API se apoyarán en una clase Java o en un método extra para almacenar los ficheros pdf.

Todos los documentos pdf generados se guardarán en la carpeta raíz del proyecto backend.

Dependencias requeridas

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

Ejemplo de generación de un documento pdf.

A continuación, os facilitamos un fragmento de un método para generar un documento pdf. Recordad que es sólo un fragmento del código y no el método completo.

```
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) Docker

Se pide entregar un fichero Dockerfile para cada uno de los proyectos a entregar, frontend y backend. Además, es necesario añadir al fichero referencias.txt una línea con el comando Docker a ejecutar para poner en funcionamiento los contenedores (una por contenedor), suponiendo que tenemos creadas las imágenes Docker con los nombres frontend:latest y backend:latest.

Sólo es necesario entregar los ficheros Dockerfile y los comandos. La corrección consistirá en comprobar la correcta definición de ambos ficheros y de los comandos de ejecución del contenedor.

Entregables

1. La carpeta que contiene ambos proyectos y la carpeta .git del proyecto. No olvidéis incluir los ficheros Dockerfile de cada proyecto.
2. Documento pdf generado por la aplicación en la raíz del proyecto backend. El nombre del fichero debe ser *info.pdf*.
3. Código de los proyectos alojado en el repositorio de Github.
4. Fichero referencias.txt correctamente cumplimentado, incluyendo los comandos Docker para ejecutar los contenedores.

Calificación del ejercicio

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

Apartado	Frontend	API	GIT	DOCKER
Puntuación	3,5	3,5	1,5	1,5

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	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.

		de funcionamiento.		
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.
Interfaz implementada mediante Vaadin 24 y gestión de las llamadas a la API.	La interfaz no se ha implementado. No usa la librería HttpRequest de Java 11.	La interfaz existe, pero no dispone de todos los campos solicitados ni es capaz de enviar las peticiones ni mostrar la información. 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.	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. Se hace uso correcto de la librería, pero existen errores de concepto en la gestión de los métodos o en los datos devueltos.	La interfaz funciona correctamente, dispone de todos los campos solicitados y permite enviar las solicitudes, así como muestra los datos correctamente. 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 ni el	Los ficheros generados son incorrectos o no coincide el nombre con el especificado. Los objetos no se	Los ficheros generados son correctos pero el nombre no coincide. Los objetos se añaden al fichero JSON	Todos los ficheros generados son correctos en formato y nombre y los datos se añaden

	fichero json solicitado.	añaden o se añaden al fichero JSON pero no son correctos.	pero no son correctos.	correctamente al fichero JSON.
DockerFile y comandos Docker	No hay ficheros Dockerfile ni se indica el comando Docker para ejecutar el contenedor que permita poner la aplicación en funcionamiento	Se han añadido unfichero Dockerfile, pero no completos y les falta información. El comando Docker no es el correcto y no funcionaría al intentar ejecutarlo.	Se han añadido los ficheros Dockerfile y tienen todas las configuraciones, pero el nombre de los ficheros jar es incorrecto o falta una definición correcta del path de ejecución. El comando Docker permite ejecutar el contenedor, pero no dispone de todas las configuraciones necesarias.	Se han añadido los ficheros Dockerfile y son completamente correctos y funcionales. El comando Docker ejecuta los contenedores correctamente y permite el funcionamiento de la aplicación.

Plantilla de cómo puede quedar el frontend.