Universidad Francisco de Vitoria UFV Madrid

NOMBRE: _____

GRUPO: _____ FECHA: _____

CALIFICACIÓN:

EXAMEN FINAL CONVOCATORIA ORDINARIA – 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 elegir el orden en que debes contestar cada apartado.

La duración de esta parte de la prueba es de 130 minutos. Y 10 minutos para la entrega.

Requisitos y pautas generales de trabajo:

1. Acceder a la URL de Github Classroom. https://classroom.github.com/a/ZM5E2DgB
2. Seguir el procedimiento de Github para la creación del nuevo repositorio.
3. Clonar el repositorio a un directorio local.
4. Se usará un único repositorio para ambos proyectos, el de front y el de back.
5. Trabajar en el enunciado del ejercicio en el repositorio en local.
6. **Es obligatorio realizar al menos un commit cada 15 minutos.**
7. Una vez dado por concluido el ejercicio, se deberá:
    1. **Realizar un commit final y generar una release con el nombre v1.0**.
    2. Comprimir la carpeta de trabajo y subirla a la tarea habilitada en el aula virtual para tal fin. Para evitar problemas de tamaño de carpeta, eliminar las carpetas node_modules y target.

Universidad Francisco de Vitoria UFV Madrid

NOMBRE: _____

GRUPO: _____ FECHA: _____

CALIFICACIÓN:

Enunciado

Se pide desarrollar una aplicación que muestre por pantalla los resultados de consultar una API externa, a través de nuestra propia API. La idea es consultar información de la API externa https://swapi.dev/ y mostrar los datos en una aplicación frontend creada con Vaadin, añadiendo en nuestra propia API información relativa a cuándo se realizó la última consulta.

Desarrollo de la aplicación

Proyecto 1: Creación de la interfaz

La página constará de dos pestañas

- **Pestaña People**: Contendrá el formulario de consulta donde rellenar el ID a consultar contra la API Swapi de los personajes de la saga.
  Una vez enviada la petición, se mostrarán por pantalla los datos obtenidos en la respuesta, incluyendo la fecha en la que se ha realizado la petición.
- **Pestaña Ships**: Contendrá el formulario de consulta donde rellenar el ID a consultar contra la API Swapi de las naves de la saga.
  Una vez enviada la petición, se mostrarán por pantalla los datos obtenidos en la respuesta, incluyendo la fecha en la que se ha realizado la petición.
- El formulario, en ambas pestañas, consistirá en un único campo numérico donde informar el id a consultar y un botón para enviar la consulta.
- Los datos se mostrarán en cada pestaña, a continuación de cada uno de los botones.
- La fecha mostrada debe tener el formato indicado a continuación: “moments ago”, “4 days ago”, “9 years ago”, etc.
  - Esto se consigue haciendo uso de la librería PrettyTime, especificada en el apartado de creación de la API.

Tips para el desarrollo:

1. Se realizará un proyecto específico para la interfaz, usando el arquetipo Vaadin.
2. La versión de Java a utilizar es **Java 11**.
3. Utilización del framework **Vaadin** 14.7.3.
4. Todas las dependencias deberán ser gestionadas por **Maven**.
5. Archetype de Vaadin para empezar la aplicación
   a. GroupID: com.vaadin
   b. Artifact ID: vaadin-archetype-application
   c. Version: 14.7.3
6. Definir el Group ID del proyecto a “ufv.extraordinaria.dis” y el Artifact ID como las iniciales del alumno en minúsculas.
7. Se valorará positivamente un diseño de la UI adecuado.

Proyecto 2: API REST

Se realizará un proyecto Springboot usando la herramienta alojada en el enlace https://start.spring.io/.
Los datos se leerán de la API mediante el desarrollo de una aplicación que permita, mediante el uso **de dos métodos GET**, obtener la lista de los datos solicitados desde cada pestaña, incluyendo la fecha en la que se ha realizado la consulta de estos.

Los datos a consultar se encuentran en una API REST externa alojada en la siguiente URL: https://swapi.dev/.

Método 1/2: GET

Este método devolverá al front un objeto JSON con los datos obtenidos de Swapi para el personaje correspondiente al id recibido y debe almacenar la hora en la que se ha realizado la consulta.

Se muestra un ejemplo de petición a la API Swapi para obtener los datos de un personaje, dado su id: ***https://swapi.dev/api/people/1.***

Método 2/2: GET

Este método devolverá al front un objeto JSON con los datos obtenidos de Swapi para la nave correspondiente al id recibido y debe almacenar la hora en la que se ha realizado la consulta.

Se muestra un ejemplo de petición a la API Swapi para obtener los datos de una nave, dado su id: ***https://swapi.dev/api/starships/2.***

Tips para el desarrollo:

- Dependencias adicionales (GSON).
  - GroupID: com.google.code.gson
  - Artifact ID: gson
  - Version: 2.6.2
- Dependencias adicionales (Prettytime, para la gestión de las fechas por pantalla).
  - GroupID: org.ocpsoft.prettytime
  - Artifact ID: prettytime
  - Version: 4.0.0.Final
- Referencia: https://www.ocpsoft.org/prettytime/
- Las fechas se almacenan en un objeto “Date”. new Date(). Puede pasarse como parámetro la fecha en milisegundos en caso de necesitar una diferente a la actual. new Date(System.currentTimeMillis() + 1000*60*10).

En esta ocasión los datos **NO** se encuentran en un fichero JSON local, sino que se deben consultar desde la API creada a la API https://swapi.dev/.

Se facilitan dos ficheros Java que contienen, cada uno, la clase correspondiente para almacenar la información obtenida desde dicha API, incluyendo un campo Date que debe rellenar el alumno como mejor estime oportuno.
Estos ficheros se encuentran en el repositorio clonado al comienzo del examen y deberán ser copiados/movidos a la carpeta correspondiente.

La conexión desde la API local a la API remota se realizará mediante HTTPRequest, incluido en Java 11.

La arquitectura del sistema final, por tanto, será de la siguiente manera:

Entregables

- Ficheros de los dos proyectos y la carpeta .git.
  - Eliminar la carpeta target y la carpeta node_modules para evitar generar ficheros comprimidos muy grandes.
- Fichero referencias.txt, en el que deben estar referenciadas todas las fuentes que hayas consultado para resolver la prueba. Sólo aquellas que has usado finalmente, no todas por las que has pasado.
- No olvidar generar una release con el contenido del proyecto y subirla al repositorio remoto clonado al comienzo del examen.

Comprime todos los estos ficheros en un archivo zip: `TuApellidoNombre.zip` que es el que deberás subir finalmente a la plataforma.

Calificación del ejercicio

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

Apartado	**Git**	**Vaadin-Front**	**API Local**	**Consulta Swapi**
Puntuación	1,5	2,5	4	2