**Prácticas TION**

SATURNINO BALLESTER RODRÍGUEZ

ANDÉS DAVID ROJAS CARRILLO

26 de enero de 2022

**Texto

Descripción generada automáticamente**

**Máster en Desarrollo de Aplicaciones y Servicios Web**

**2021 – 2022**

Índice

[Introducción 3](#_Toc94116749)

[Cuerpo del trabajo 3](#_Toc94116750)

[Práctica 1: Introducción a Java 3](#_Toc94116751)

[Introducción 3](#_Toc94116752)

[Instalación de IDE 3](#_Toc94116753)

[Actividades 4](#_Toc94116754)

[Conclusiones 8](#_Toc94116755)

# Introducción

# Cuerpo del trabajo

## Práctica 1: Introducción a Java

### Introducción

Se va a realizar la preparación de los equipos de trabajo para el entorno de desarrollo necesario para realizar los futuros proyectos que se llevarán a cabo en este campo de trabajo.

Para el desarrollo de esta primera parte, se utilizará IntelliJ IDEA Ultimate.

Creando un proyecto que incorpore Maven, conseguiremos una herramienta para facilitar el proceso de definición de las etapas del ciclo de vida o de la realización de pruebas unitarias.

### Instalación de IDE

Se ha Instalado el entorno de desarrollo para crear proyectos Java, en este caso se ha tomado la decisión de utilizar IntelliJ IDEA.

Imagen que contiene Gráfico de burbujas

Descripción generada automáticamente

Ilustración . IDE instalado para el desarrollo

### Actividades

1. **Convierte el código del proyecto Biblioteca a Maven.**

Primero creamos el proyecto de java con Maven y se introduce el código dado en el siguiente enlace: <https://github.com/hibernator11/Biblioteca-java>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Creación de un proyecto MAVEN en IntelliJ IDEA

Texto

Descripción generada automáticamente

Ilustración . Creación de un proyecto MAVEN en IntelliJ IDEA (2)

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración . Importación de las clases en el proyecto creado

1. **Añade un método para convertir el catálogo a formato JSON**

A continuación, se añaden las dependencias del *org.json* para importar la librería de tratamiento de JSON:

Texto

Descripción generada automáticamente

Ilustración . Creación del apartado de dependencias en el POM.xml

Posteriormente, se incluirá la librería en la clase *Catalogo.java*:

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Ilustración . Importación de la librería JSONArray de org.json en Catalogo.java

Al mismo tiempo, se importará en *Publicacion.java, Libro.java, Periodico.java*; la librería de objeto.

Texto

Descripción generada automáticamente

Ilustración . Importación de la librería JSONObject en las clases Publicacion, Libro y Periodico

Tras realizar las importaciones pertinentes, se modificará la clase *Publicacion* a *abstract* y crear así el método *getJSON():*

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración . Modificación de la clase Publicacion.java

Una vez se implementa el método en la clase *Publicación*, se procederá a implementar el mismo en las clases *Libro* y *Periodico:*

Texto

Descripción generada automáticamente

Ilustración . Implementación de getJSON en Libro

Texto

Descripción generada automáticamente

Ilustración . Implementación de getJSON en Periodico

Se el método de *crearJSON()*:

Texto

Descripción generada automáticamente

Ilustración . Creación del método crearJSON

Finalmente, se llama en el *main* de *Catalogo.java* para invocar el método creado:

Texto

Descripción generada automáticamente

Ilustración . Invocación de métodos en el main.java

Para concluir, se ejecutará el proyecto para obtener la salida correspondiente.

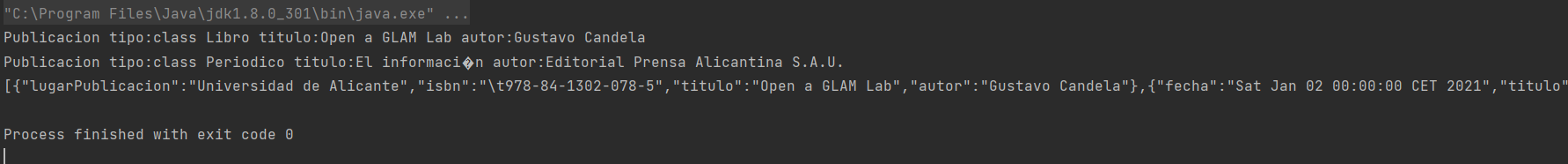


Ilustración . Ejecución y salida del proyecto

Se puede observar que el proyecto se ha ejecutado correctamente y no ha tenido problemas en su ejecución.

# Conclusiones

conclusiones.