



Aplicación Modelo-Vista-Controlador con funcionalidad CRUD

Model-View-Controller application with CRUD functionality
Aplicación Modelo-Vista-Controlador con funcionalidad CRUD

Bryan Steven Pardo Nuñez,
Thomas Espitia Lopez
bspardo@poligran.edu.co
jpenata@poli.edu.co
Politécnico Gran Colombiano
Colombia

Resumen

Este proyecto tiene como objetivo desarrollar una aplicación en **Java** basada en la arquitectura **Modelo-Vista-Controlador (MVC)** para optimizar la gestión de insumos agrícolas en una empresa dedicada al cultivo de hortalizas. Actualmente, el control de fertilizantes, semillas y pesticidas se realiza manualmente, lo que genera errores y dificulta la trazabilidad del inventario.

Para solucionar este problema, se implementará un sistema **CRUD (Crear, Leer, Actualizar y Eliminar)** que permitirá registrar, consultar, modificar y eliminar información sobre los insumos. Esto facilitará una administración centralizada, mejorará la toma de decisiones y optimizará el control del inventario dentro del proceso productivo.

El desarrollo del proyecto se enfocará en la implementación de la aplicación, asegurando un diseño eficiente y buenas prácticas de programación orientada a objetos en Java.

Palabras clave

modelo-vista-controlador (MVC), gestión de inventario, insumos agrícolas, programación orientada a objetos (POO), java collection framework, optimización de procesos.

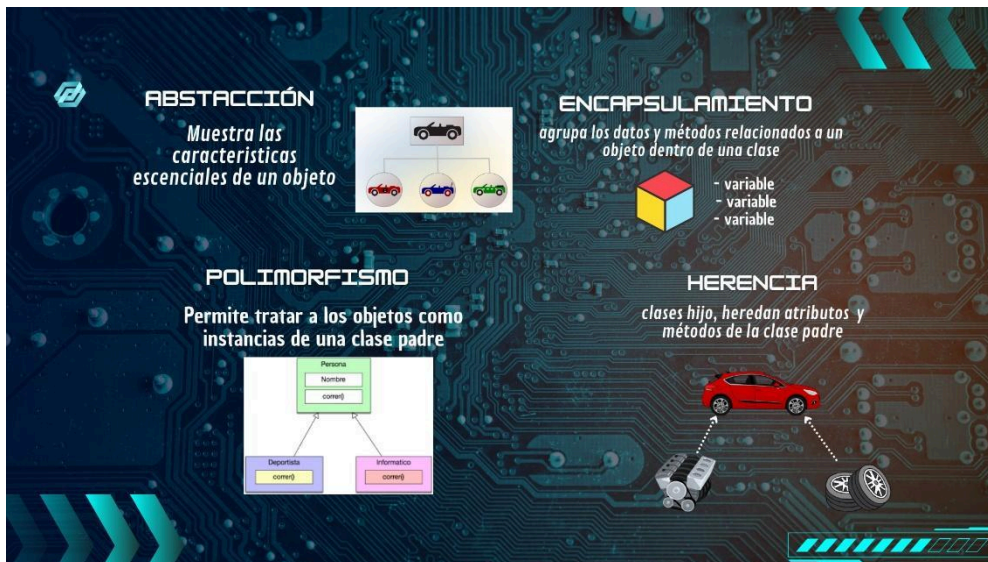
INTRODUCCIÓN

En la actualidad, la tecnología juega un papel fundamental en la optimización de procesos dentro del sector agrícola. La digitalización de la gestión de insumos permite mejorar la eficiencia operativa y minimizar errores en el control del inventario. Implementar soluciones informáticas en este ámbito no solo facilita el acceso a la información en tiempo real, sino que también contribuye a una mejor toma de decisiones.

Este proyecto busca desarrollar una aplicación que permita administrar de manera centralizada los insumos agrícolas mediante un sistema de registro estructurado. Utilizando **JavaFX** y el patrón **Modelo-Vista-Controlador (MVC)**, la aplicación ofrecerá una interfaz intuitiva y funcional que agilice el manejo de datos y optimice la trazabilidad de los productos.

A lo largo del desarrollo, se aplicarán conceptos clave de **programación orientada a objetos (POO)** y se hará uso de **estructuras de datos** eficientes para garantizar el correcto almacenamiento y manipulación de la información. De esta manera, se propone una solución tecnológica que responda a las necesidades de automatización en la gestión de inventarios agrícolas.

Pilares Programación Orientada a Objetos (POO)



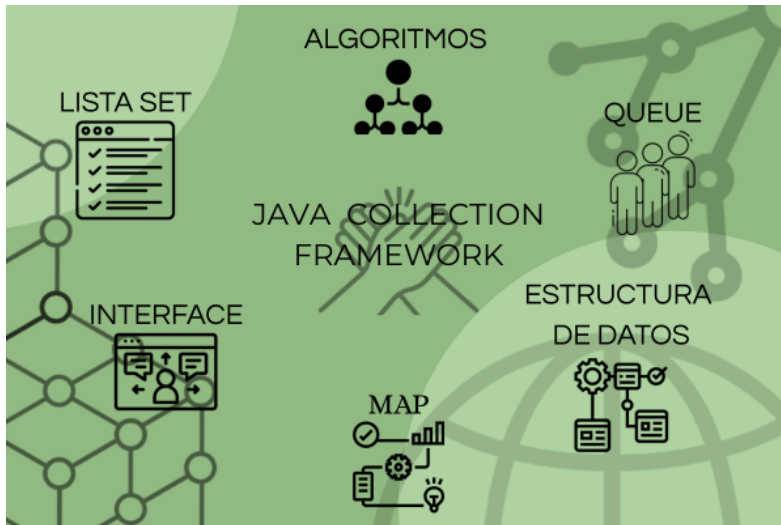
Arquitectura Modelo-Vista-Controlador



Repositorio



Java Collection Framework

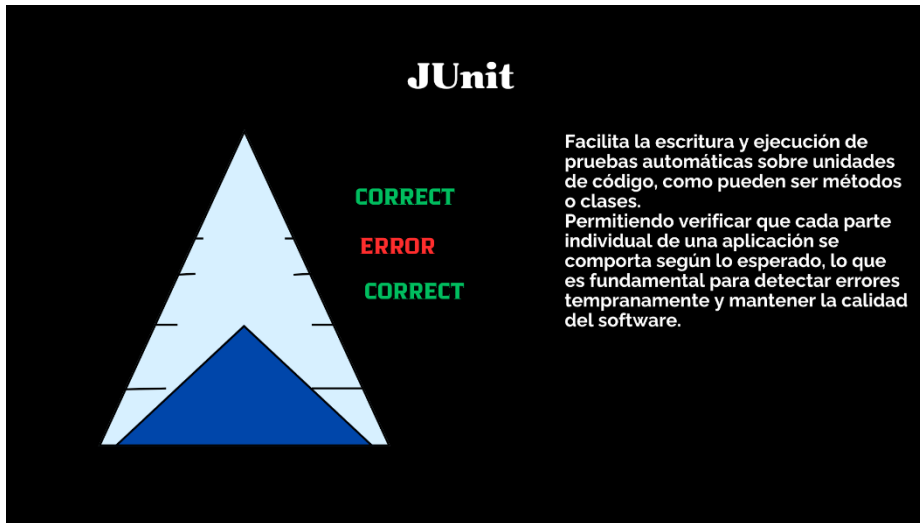


Javadoc

```
/** Descripción detallada del método
 * @param se usa para los parámetros
 */

public void metodo() {
    codigo
}
```

JUnit



Enunciado GPT Prompt

En una pequeña granja, se registra de manera manual la llegada y el uso de distintas variedades de **semillas** para los cultivos, lo que ocasiona errores, pérdida de información y dificultades en la planificación de la plantación. Debido a la diversidad de semillas y su importancia en la producción, es crucial contar con un sistema que permita gestionarlas de forma ordenada y precisa.

Se requiere desarrollar una aplicación en JavaFX que implemente un único CRUD para el registro de **semillas**, que permita:

- **Crear:** Registrar nuevas semillas ingresando un identificador único, el nombre o variedad, la cantidad disponible y, opcionalmente, su origen o fecha de cosecha.
- **Leer:** Consultar y visualizar la lista de semillas registradas, facilitando una rápida revisión del inventario.
- **Actualizar:** Modificar la información de una semilla existente, en caso de requisitos de reposición o ajustes en la cantidad.
- **Eliminar:** Quitar aquellos registros que ya no sean relevantes o que se hayan ingresado por error.

Prototipo Vista de la Aplicación Para el Usuario

ID	<input type="text"/>	<table><tr><th>ID</th><th>Nombre</th><th>Nombre</th><th>Cant. Disponible</th><th>Tiem. Cosecha</th><th>Origen</th></tr><tr><td colspan="6">No content in table</td></tr></table>	ID	Nombre	Nombre	Cant. Disponible	Tiem. Cosecha	Origen	No content in table					
ID	Nombre	Nombre	Cant. Disponible	Tiem. Cosecha	Origen									
No content in table														
Nombre	<input type="text"/>													
Cantidad Disponible	<input type="text"/>													
Tiempo de Cosecha	<input type="text"/>													
Origen	<input type="text"/>													
Tipo	<input type="text"/>													

Crear

Leer

Actualizar

Eliminar

Leer Todo

Serializar

Deserializar

Cronograma de Actividades

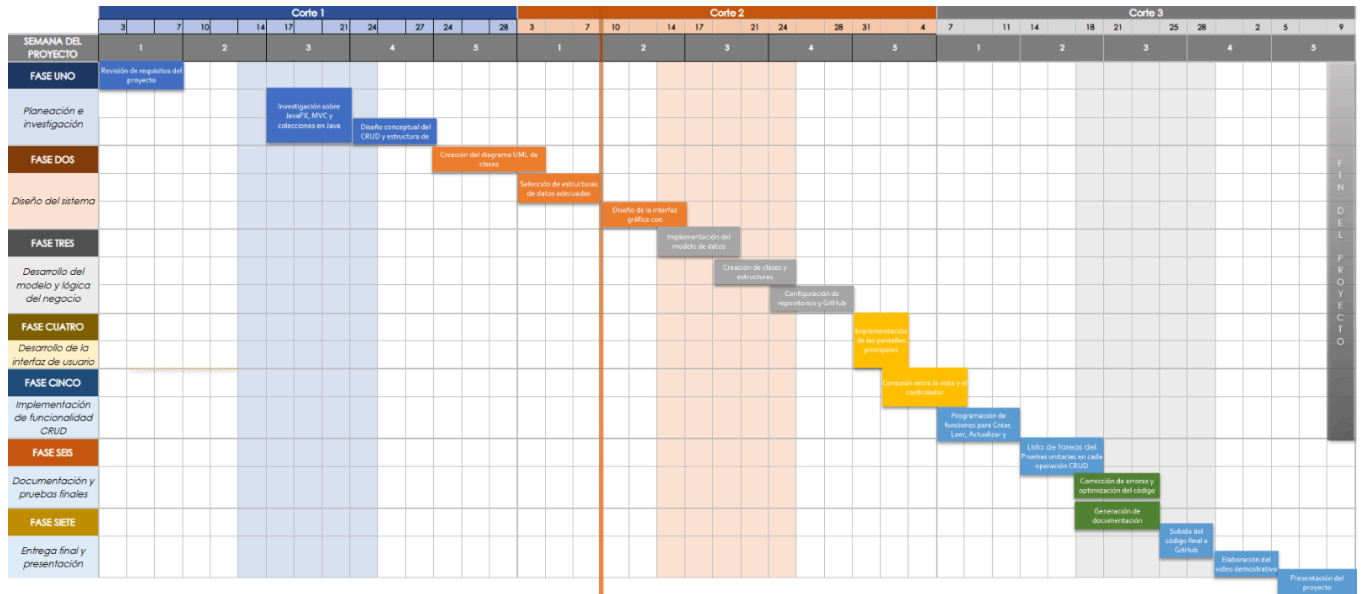


Diagrama De clases UML

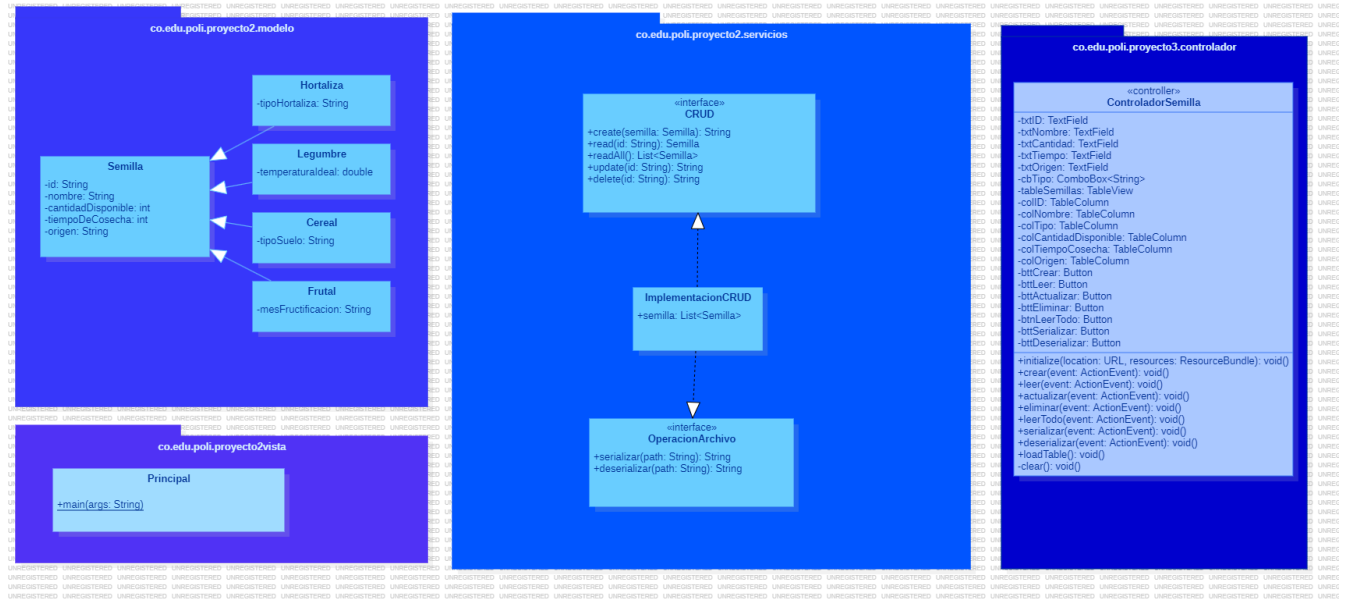
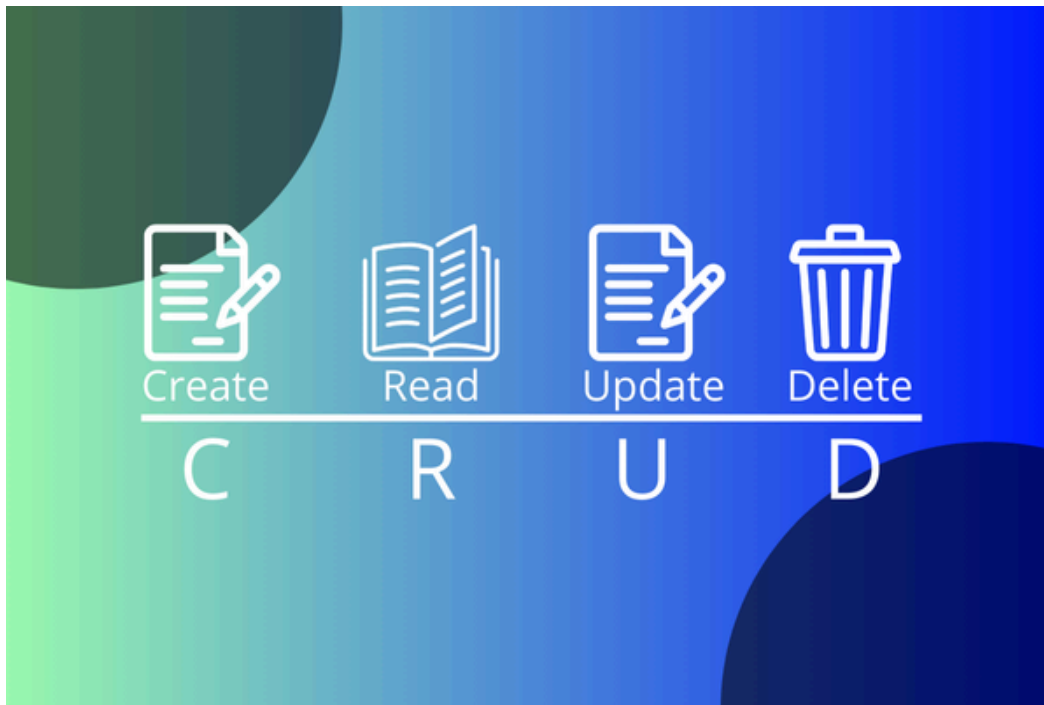


Gráfico Operaciones CRUD

CRUD



Create



CREATE

```

public String create(Semilla semilla) {
    if (count == semillas.length) {
        expandArray();
    }
    semillas[count++] = semilla;
    return "Semilla creada exitosamente.";
}

```



Read

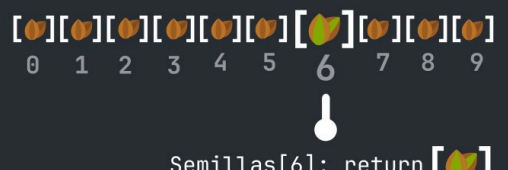


READ

```

public Semilla read(String id) {
    for (int i = 0; i < count; i++) {
        if (semillas[i].getId().equals(id)) {
            return semillas[i];
        }
    }
    return null;
}

```



Semillas[6]; return [🌱]

Update

Update

modifica los datos existentes en una base de datos o en un sistema de almacenamiento persistente



Base de
datos



Formulario
de Update



Base de
datos

UPDATE

```
public String update(Semilla semilla) {  
    for (int i = 0; i < count; i++) {  
        if (semillas[i].getId().equals(semilla.getId())) {  
            semillas[i] = semilla;  
            return "Semilla actualizada exitosamente.";  
        }  
    }  
    return "No se encontró la semilla a actualizar.";  
}
```



Delete

Delete

borra datos de una base de datos o un sistema de almacenamiento



Base de datos

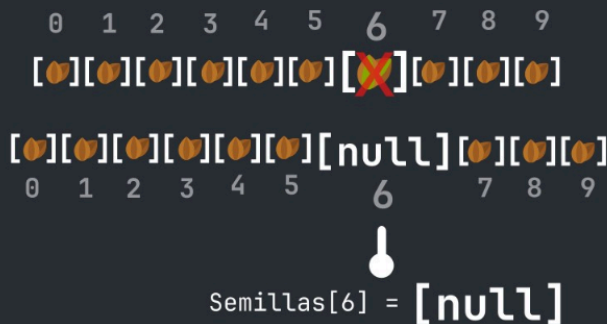
(quitar registro)



Registro eliminado

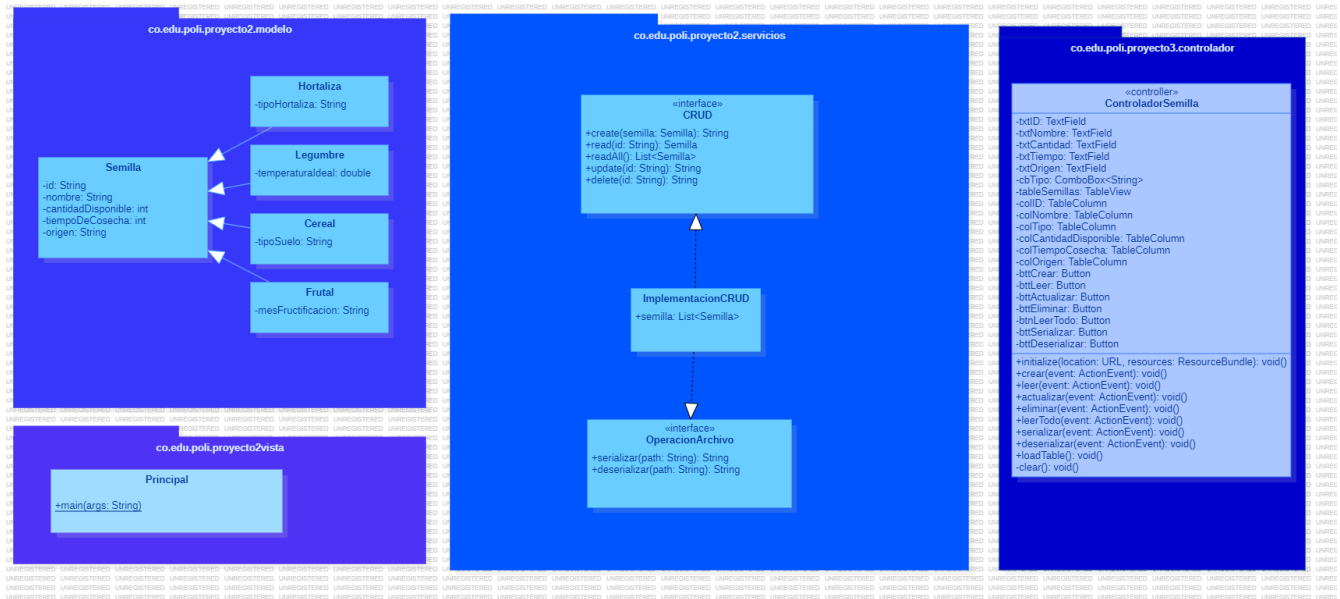
DELETE

```
public String delete(String id) {  
    for (int i = 0; i < count; i++) {  
        if (semillas[i].getId().equals(id)) {  
            for (int j = i; j < count - 1; j++) {  
                semillas[j] = semillas[j + 1];  
            }  
            semillas[--count] = null;  
            return "Semilla eliminada exitosamente."  
        }  
    }  
    return "No se encontró la semilla a eliminar."  
}
```



Entrega 3:

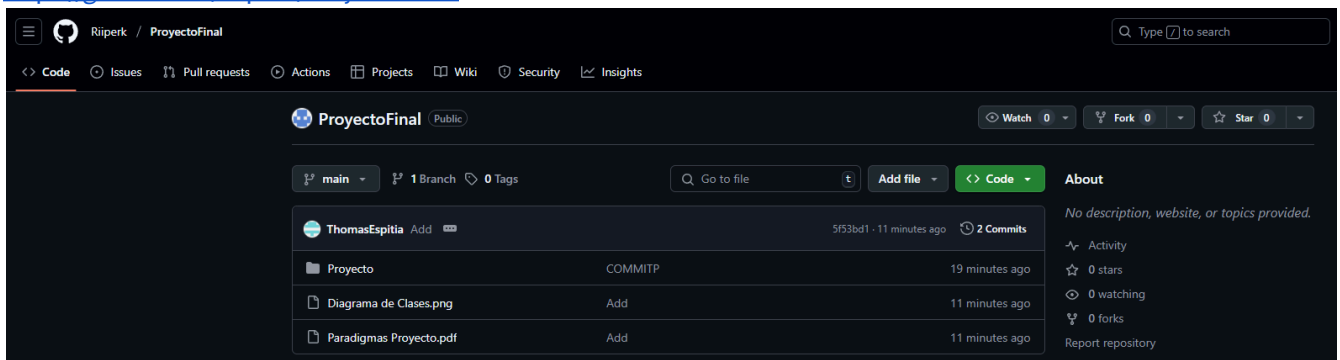
Diagrama de clases:



Link del Repositorio en GitHub

En el siguiente enlace se encontrará el repositorio de GitHub con el respectivo código del diagrama UML, tomando en cuenta el enunciado y usando lo visto en clase (Método CRUD, Serializar y deserializar) siendo presentado en un menú para el usuario.

<https://github.com/Riiperk/ProyectoFinal>



Versión final interfaz gráfica:

[illegible]

Link del video mostrando el producto final

El desarrollo Link que dirige al video explicando y mostrando la funcionalidad del CRUD con interfaz gráfica:

<https://www.youtube.com/watch?v=OOhr443LfCo>

DISCUSIÓN Y CONCLUSIÓN

El desarrollo de esta aplicación responde a la necesidad de mejorar la gestión de insumos agrícolas mediante la digitalización del inventario. La implementación del **CRUD** facilita el control sobre semillas y sus diferentes variaciones reduciendo errores y optimizando la toma de decisiones.

Uno de los principales desafíos ha sido garantizar una estructura eficiente del sistema, para lo cual se ha aplicado el **patrón Modelo-Vista-Controlador (MVC)** y el uso de **colecciones en Java**, asegurando un manejo óptimo de los datos. Además, se han integrado **pruebas** para validar la funcionalidad del sistema y evitar fallos en la manipulación de la información.

REFERENCIAS BIBLIOGRÁFICAS

DataCamp. (2025). *Abstracción Java*. Recuperado el 26 de abril de 2025, de <https://www.datacamp.com/es/doc/java/abstraction>.

DataCamp. (2025). *Encapsulación Java*. Recuperado el 26 de abril de 2025, de <https://www.datacamp.com/es/doc/java/encapsulation>