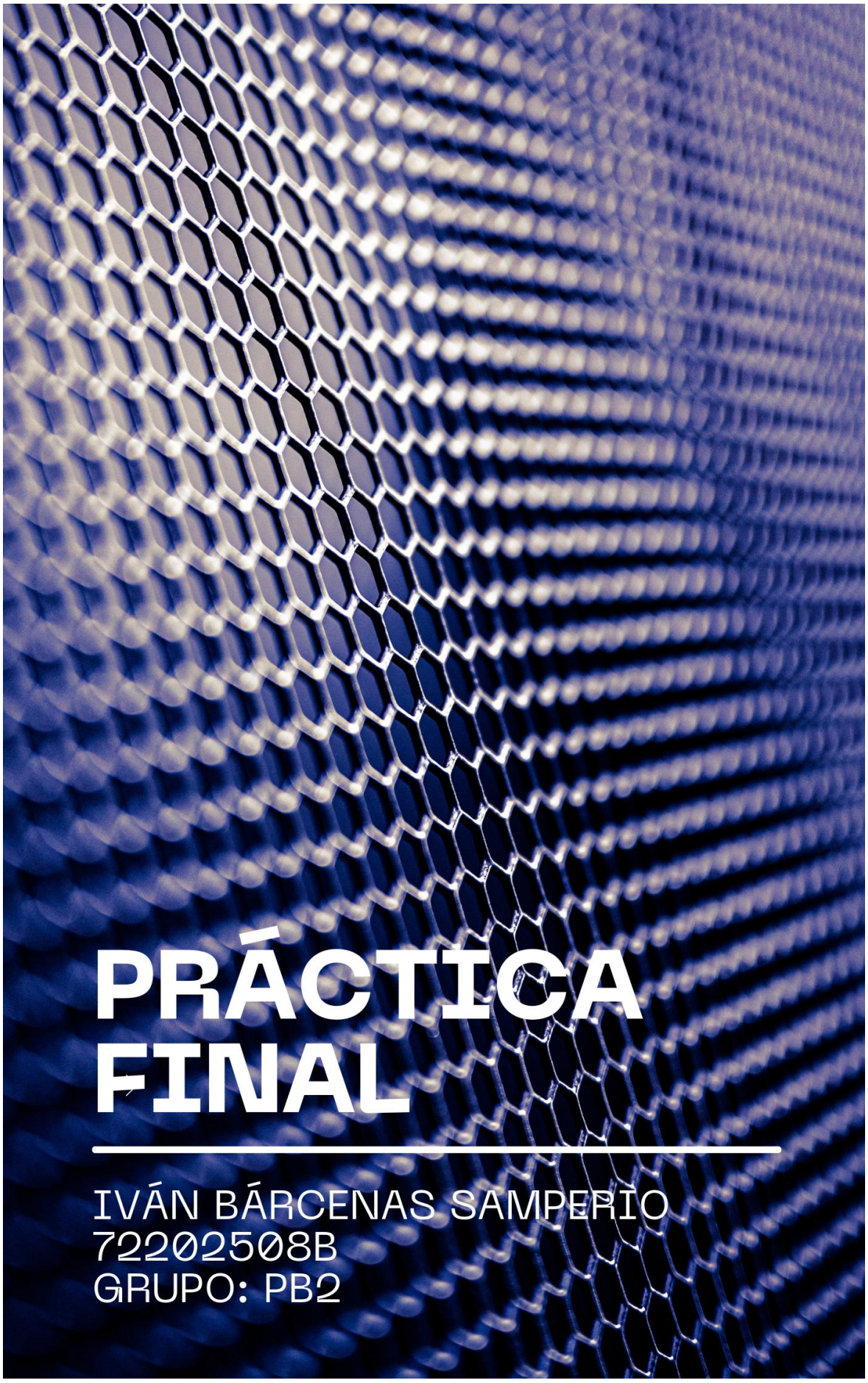


PRÁCTICA FINAL

IVÁN BÁRCENAS SAMPERIO
72202508B
GRUPO: PB2

Funcionalidad 1

Demostración de la utilización de los parámetros utilizados en la ejecución del código:

.vscode > launch.json > ...

```jsonc
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "Current File",
            "request": "launch",
            "mainClass": "${file}"
        },
        {
            "type": "java",
            "name": "App",
            "request": "launch",
            "mainClass": "app.App",
            "projectName": "Practica-Final_704dbc09",
            "args": "--repository bin"
        }
    ]
}
```

Utilizo un launch.json para facilitar la tarea de ejecución.

```java
public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        IRepository repository;
        BaseView view;

        if(args.length == 4){
            repository = new NotionRepository(args[2],args[3]);
            view = new InteractiveView();
        }else{
            repository= new BinaryRepository();
            view = new InteractiveView();
        }

        Model model = new Model(repository);
        Controller c = new Controller(model,view);

        c.start();
        c.end();
    }
}
```

Dentro del main establecemos el repositorio que deseamos utilizar dependiendo de los parámetros que se hayan establecido. Si se han establecido 4 parámetros significa que queremos utilizar el repositorio Notion, de otra forma establecemos el repositorio binario haciendo que este sea el predeterminado.

Establecemos también la vista interactiva, no importa el repositorio utilizado ya que es la única vista implementada.

Por último, comenzamos la ejecución del código llamando al método start del controlador, cuando se desee salir de la ejecución automáticamente pasará al método end, dónde se cerrará el código de forma ordenada.

Funcionalidad operaciones CRUD

Start

```
public void start(){
    if(model.loadData()){
        view.showMessage(msg:"Datos cargados correctamente.");
    }else{
        view.showErrorMessage(msg:"No se encontro el fichero o no se han podido cargar los datos correctamente");
    }
    view.init();
}
```

Se llama al método start, dónde primero se comprueba que se cargan los datos de las anteriores ejecuciones de forma correcta. Si es así mostrará un mensaje indicándolo, de no ser así mostrará un mensaje de error.

Acto seguido se llamará al método init de la vista donde comenzaría la funcionalidad del código.

loadData

```
public boolean loadData(){
    if (ficheroSerializado.exists() && ficheroSerializado.isFile()) {
        ObjectInputStream ois = null;
        try {
            ois = new ObjectInputStream(new FileInputStream(ficheroSerializado));
            ArrayList<Task> listado = (ArrayList<Task>) ois.readObject();
            for (Task task : listado) {
                repository.addTask(task);
            }
        } catch (IOException | ClassNotFoundException ex) {
            System.err.println("Error durante la deserializacion: " + ex.getMessage());
            return false;
        } finally {
            if (ois != null) {
                try {
                    ois.close();
                } catch (IOException ex) {
                    System.err.println("Error durante la deserializacion: " + ex.getMessage());
                    return false;
                }
            }
        }
        return true;
    } else {
        return false;
    }
}
```

El método loadData carga las tareas del fichero binario que se utiliza para guardar el estado de anteriores ejecuciones.

En la siguiente foto se muestra como se cargan correctamente las tareas guardadas en ejecuciones anteriores.

```
Datos cargados correctamente.

|--------------------------------------|

|----------| MENU TAREAS |------------|

|--------------------------------------|

| 1) Añadir una tarea                  |

| 2) Listado de las tareas             |

| 3) Modificar una tarea               |

| 4) Importar/Exportar tareas          |

| 5) Salir del programa                |

|--------------------------------------|
Introduzca una opcion: 2

|-----------------------------------------|

|----------| SUBMENU LISTADO |------------|

|-----------------------------------------|

| 1) Ordenado por prioridad (Sin completar)|

| 2) Listado completo                     |

| 3) Salir del submenu                    |

|-----------------------------------------|
Introduzca una opcion: 2
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
```

BaseView

```java
package View;

import controller.Controller;

public abstract class BaseView {

    protected Controller controller;

    public void setController(Controller c){
        this.controller=c;
    }
    public abstract void init();
    public abstract void showMessage(String msg);
    public abstract void showErrorMessage(String msg);
    public abstract void end();

}
```

La clase abstracta BaseView, en donde se encuentran los cuatro métodos abstractos que han de heredar cada vista que se relacione con ella. En este caso solo la vista interactiva.

```java
public class InteractiveView extends BaseView{
```

```java
@Override
public void showMessage(String msg) {
    System.out.printf(format:"%s\n",msg);

}

@Override
public void showErrorMessage(String msg) {
    System.out.printf(format:"ERROR. %s\n",msg);
}

@Override
public void end() {
    showMessage(msg:"Saliendo...");
}
```

En la anterior foto vemos tres de los métodos implementados:

- showMessage: Muestra un mensaje que se le pasa como parámetro.
- showErrorMessage: Muestra un mensaje de error que se le pasa como parámetro.
- End: Muestra un mensaje indicando que la ejecución ha terminado.

Init

```java
@Override
public void init() {
    int opcion;
    do{
        System.out.println(x:"\n|--------------------------------------|");
        System.out.println(x:"\n|----------| MENU TAREAS |-----------|");
        System.out.println(x:"\n|--------------------------------------|");
        System.out.println(x:"\n| 1) Añadir una tarea                  |");
        System.out.println(x:"\n| 2) Listado de las tareas             |");
        System.out.println(x:"\n| 3) Modificar una tarea               |");
        System.out.println(x:"\n| 4) Importar/Exportar tareas          |");
        System.out.println(x:"\n| 5) Salir del programa                |");
        System.out.println(x:"\n|--------------------------------------|");
        System.out.printf(format:"Introduzca una opcion: ");
        opcion = scan.nextInt();
        scan.nextLine();

        switch (opcion) {
            case 1:
                addTarea();
                break;
            case 2:
                listadoTarea();
                break;
            case 3:
                subMenuModificarTarea();
                break;
            case 4:
                subMenuExportarImportar();
                break;
            case 5:
                System.out.println(x:"Saliendo...");
                break;
            default:
                System.out.println(x:"ERROR. OPCION NO VALIDA");
                break;
```

Muestra el menú con las diferentes posibilidades. Llama a un método concreto que iniciara la lógica utilizada para la opción seleccionada.

addTarea

Pide los datos al usuario y los envía al modelo a través del controlador. Si se añade de forma correcta se muestra un mensaje que lo indica y si no es así también muestra un mensaje de error.

```java
if(controller.addTarea(id,title,date,content,priority,estimatedDuration,completed)){
    showMessage(msg:"Se ha añadido la tarea correctamente.");
}else{
    showErrorMessage(msg:"Ya hay una tarea con ese identificador.");
}
```

```java
public boolean addTarea(int id, String title, String date, String content, int priority, int estimatedDuration,boolean completed) {
    return model.addTarea(id,title,date,content,priority,estimatedDuration,completed);

}
```

```java
public boolean addTarea(int id, String title, String date, String content, int priority, int estimatedDuration,boolean completed) {
    Task tareaActual= new Task(id, priority, estimatedDuration, title, content, date, completed);
    if(repository.addTask(tareaActual)!=null){
        return true;
    }else{
        return false;
    }
}
```

En el modelo crea la tarea mediante el constructor utilizado en la clase tarea y la envía al repositorio seleccionado previamente (más tarde veremos en profundidad ambos repositorios), si esta se añade de forma correcta devuelve hasta la vista mediante el controlador.

Demostración de su correcto funcionamiento:

```
Introduzca una opcion: 1
Introduzca el identificador unico de la tarea: 8945
Introduzca el titulo de la tarea: Demostrar funcionamiento
Introduzca la fecha de la tarea.
Introduzca el anio: 2024
Introduzca el mes: 12
Introduzca el dia: 11
Introduzca el contenido de la tarea: Demostrar que funciona correctamente
Introduzca la prioridad de la tarea(1-5)(5 maxima prioridad): 1
Introduzca la duracion estimada de la tarea en minutos: 2
Introduzca si la tarea esta completada(y/n): y
Se ha añadido la tarea correctamente.
```

listadoTarea

Este método nos lleva a un submenú que nos muestra las opciones de listados que tenemos:

- Ordenado por prioridad (tareas sin completar)
- Listado de todas las tareas

```java
private void showListadoPrioridad() {
    ArrayList<String> listado = controller.getListadoPrioridad();
    for (String taskActual : listado) {
        System.out.println(taskActual);
    }
}

private void showListadoCompleto() {
    ArrayList<String> listado = controller.showListadoCompleto();
    for (String taskActual : listado) {
        System.out.println(taskActual);
    }
}
```

En ambas opciones se obtiene un ArrayList de cadenas de caracteres con las correspondientes tareas, desde el modelo a través del controlador.

```java
public ArrayList<String> getListadoPrioridad() {
    return model.getListadoPrioridad();
}

public ArrayList<String> showListadoCompleto() {
    return model.showListadoCompleto();
}
```

```java
public ArrayList<String> getListadoPrioridad() {
    ArrayList<Task> listado = repository.getAllTask();
    ArrayList<String> listadoPrioridad = new ArrayList<>();
    for(int i=5; i>=0; i--){
        for (Task task : listado) {
            if((task.getPriority()==i)&&(!task.isCompleted())){
                String taskActual = task.toString();
                listadoPrioridad.add(taskActual);
            }
        }
    }
    return listadoPrioridad;
}

public ArrayList<String> showListadoCompleto() {
   ArrayList<Task> listado = repository.getAllTask();
   ArrayList<String> listadoStrings = new ArrayList<>();
   for (Task task : listado) {
        String taskActual = task.toString();
        listadoStrings.add(taskActual);
    }
    return listadoStrings;
}
```

En el modelo obtenemos el listado completo de las tareas gracias al método del repositorio getAllTask.

- Ordenador por prioridad: Creamos un ArrayList de cadenas de caracteres que más tarde devolveremos con todas aquellas tareas que no han sido completadas y están ordenadas por prioridad

```
|-----------------------------------------|

|----------| SUBMENU LISTADO |------------|

|-----------------------------------------|

| 1) Ordenado por prioridad (Sin completar)|

| 2) Listado completo                      |

| 3) Salir del submenu                     |

|-----------------------------------------|
Introduzca una opcion: 1
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
```

- Listado completo: Creamos un ArrayList de cadenas de caracteres que más tarde devolveremos con todas las tareas.

```
|----------------------------------------|

|----------| SUBMENU LISTADO |-----------|

|----------------------------------------|

| 1) Ordenado por prioridad (Sin completar)|

| 2) Listado completo                    |

| 3) Salir del submenu                   |

|----------------------------------------|
Introduzca una opcion: 2
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
4958 | Imprimir apuntes |  3/12/2023 | Imprimir apuntes para el examen final | 2 | 20 | true |
```

subMenuModificarTarea

Este método nos lleva a un submenú que nos muestra las opciones de modificar que tenemos:

- ### Modificar si la tarea está completa o incompleta

  Solicitamos el identificador de la tarea que queremos solicitar y según el estado de la tarea o si la tarea existe muestra un mensaje.

```java
private void modificarCompleted() {
    System.out.printf(format:"Introduzca el identificador de la tarea que desee modificar: ");
    int id = scan.nextInt();
    scan.nextLine();

    if(controller.modificarCompleted(id)==1){
        showMessage(msg:"Tarea marcada como completada");
    }else if(controller.modificarCompleted(id)==2){
        showMessage(msg:"Tarea marcada como no completada");
    }else if(controller.modificarCompleted(id)==0){
        showErrorMessage(msg:"El identificador no pertenece a ninguna tarea.");
    }
}
```

  Enviamos el identificador al modelo a traves del controllador y sera este el que le devuelva el estado de la tarea a la vista.

```java
public int modificarCompleted(int id) {
    int estado = model.modificarCompleted(id);
    if(estado==1){
        return 1;
    }else if(estado == 2){
        return 2;
    }else{
        return 0;
    }
}
```

En el modelo recibimos todas las tareas con la funcion del repositorio getAllTasks. Cuando encuentra la tarea seleccionada hace el cambio.

```
public int modificarCompleted(int id) {
    ArrayList<Task> listado = repository.getAllTask();
    for (Task task : listado) {
        if(task.getIdentifier()==id){
            if(task.isCompleted()){
                Task taskActualizada = new Task(id, task.getPriority(), task.getEstimatedDuration(), task.getTitle(), task.getContent(), task.getDate(), completed:false);
                repository.modifyTask(taskActualizada);
                return 2;
            }else{
                Task taskActualizada = new Task(id, task.getPriority(), task.getEstimatedDuration(), task.getTitle(), task.getContent(), task.getDate(), completed:true);
                repository.modifyTask(taskActualizada);
                return 1;
            }
        }
    }
    return 0;
}
```

Demostración de su correcto funcionamiento:

```
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
```

```
|-----------------------------------------|

|----------| SUBMENU MODIFICAR |-----------|

|-----------------------------------------|

| 1) Marcar como completa / incompleta    |

| 2) Modifiar informacion                 |

| 3) Eliminar tarea                       |

| 4) Salir del submenu                    |

|-----------------------------------------|
Introduzca una opcion: 1
Introduzca el identificador de la tarea que desee modificar: 5738
Tarea marcada como completada
```

```
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | true |
```

- ## Modificar el resto de información

Solicitamos al usuario el identificador de la tarea y los nuevos parámetros. Enviamos los datos al modelo a través del controlador. En función de si se realiza la operación correctamente o no mostramos un mensaje que lo indique.

```
if(controller.modificarTarea(id,title,date,content,priority,estimatedDuration,completed)){
    showMessage(msg:"Tarea modificada correctamente.");
}else{
    showErrorMessage(msg:"El identificador no pertenece a ninguna tarea.");
}
```

```
public boolean modificarTarea(int id, String title, String date, String content, int priority,int estimatedDuration, boolean completed) {
    return model.modificarTarea(id,title,date,content,priority,estimatedDuration,completed);
}
```

En el modelo obtendremos todas las tareas y cuando encontremos la que queremos modificar crearemos la nueva tarea y la enviaremos como parámetro al método

modifyTask del repositorio para que intercambie los parámetros.

```
public boolean modificarTarea(int id, String title, String date, String content, int priority, int estimatedDuration, boolean completed) {
    ArrayList<Task> listado = repository.getAllTask();
    for (Task task : listado) {
        if(task.getIdentifier()==id){
            Task taskActualizada = new Task(id, priority, estimatedDuration, title, content, date, completed);
            repository.modifyTask(taskActualizada);
            return true;
        }
    }
    return false;
}
```

Demostración de su correcto funcionamiento:

```
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
Introduzca una opcion: 2
Introduzca el identificador de la tarea que desee modificar: 5738
Introduzca el nuevo titulo de la tarea: Hacer fotos
Introduzca la fecha de la tarea.
Introduzca el anio: 2022
Introduzca el mes: 6
Introduzca el dia: 12
Introduzca el nuevo contenido de la tarea: Hacer fotos de diferentes paisajes
Introduzca la nueva prioridad de la tarea(1-5)(5 maxima prioridad): 1
Introduzca la nueva duracion estimada de la tarea en minutos: 49
Introduzca si la tarea esta completada(y/n): n
Tarea modificada correctamente.
5738 | Hacer fotos | 12/ 6/2022 | Hacer fotos de diferentes paisajes | 1 | 49 | false |
```

- ## Eliminar una tarea

Solicitamos al usuario el identificador de la tarea y se lo enviamos al modelo a través del controlador. Si se elimina la tarea correctamente el modelo nos lo notifica a través del controlador y lo imprimiremos con un mensaje, de no ser así también nos lo notificará.

```
private void removeTarea() {
    System.out.printf(format:"Introduzca el identificador de la tarea que desee eliminar: ");
    int id = scan.nextInt();
    scan.nextLine();

    if(controller.removeTarea(id)){
        showMessage(msg:"Tarea eliminada correctamente.");
    }else{
        showErrorMessage(msg:"El identificador no pertenece a ninguna tarea.");
    }
}
```

```
public boolean removeTarea(int id) {
    return model.removeTarea(id);
}
```

En el método obtendremos todas las tareas como ya hemos hecho anteriormente. Compararemos los identificadores hasta encontrar la que estábamos buscando y

llamaremos al método del repositorio removeTask.

```
public boolean removeTarea(int id) {
    ArrayList<Task> listado = repository.getAllTask();
    for (Task task : listado) {
        if(task.getIdentifier()==id){
            repository.removeTask(task);
            return true;
        }
    }
    return false;
}
```

Demostracion de su funcionamiento:

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
```

```
Tarea eliminada correctamente.
```

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
```

Funcionalidad Exportar/Importar

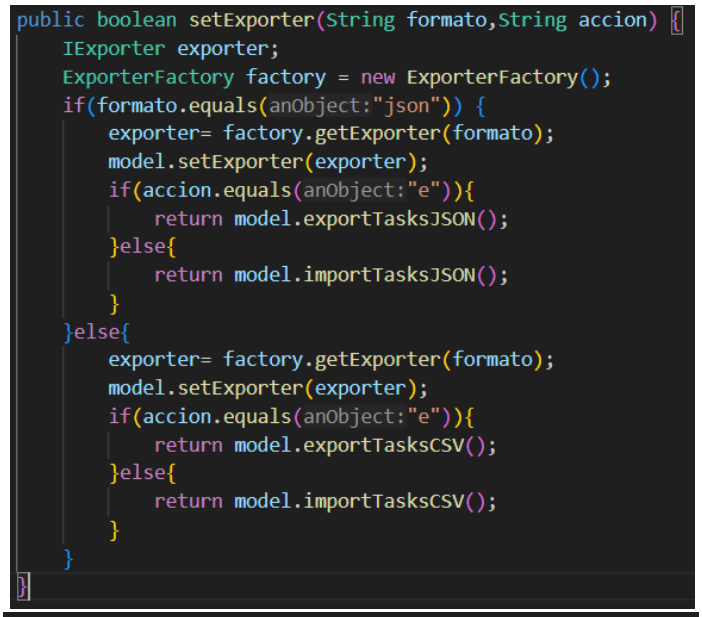
La opción de exportar o importar en el menú principal nos lleva a un submenú de exportación o importación donde se elige cual de las dos funciones quieres realizar. Una vez seleccionada la opción se preguntará al usuario que tipo de formato desea utilizar (JSON o CSV).

```
private void importTask() {
    System.out.printf(format:"Introduzca el formato de importacion (csv-json)");
    String formato = scan.nextLine();
    if(formato.equals(anObject:"csv")||formato.equals(anObject:"json")){
        if(controller.setExporter(formato,accion:"i")){
            showMessage(msg:"Importacion exitosa");
        }else{
            showErrorMessage(msg:"La importacion no ha sido exitosa");
        }
    }else{
        System.out.println(x:"ERROR. FORMATO NO VALIDO");
    }
}

private void exportTask() {
    System.out.printf(format:"Introduzca el formato de exportacion (csv-json)");
    String formato = scan.nextLine();
    if(formato.equals(anObject:"csv")||formato.equals(anObject:"json")){
        if(controller.setExporter(formato,accion:"e")){
            showMessage(msg:"Exportacion exitosa");
        }else{
            showErrorMessage(msg:"La exportacion no ha sido exitosa");
        }
    }else{
        System.out.println(x:"ERROR. FORMATO NO VALIDO");
    }
}
```

Una vez seleccionado el formato se establecera el exporter adecuado de la siguiente manera:

- Se llamará al método setExporter en el controlador y se le pasarán como parámetros el formato elegido y si se quiere exportar una “e” o si se quiere importar una “i”.
- En el método setExporter se creará tanto el exporter como una nueva clase FactoryExporter de donde obtendremos el exporter indicado en función del formato seleccionado.



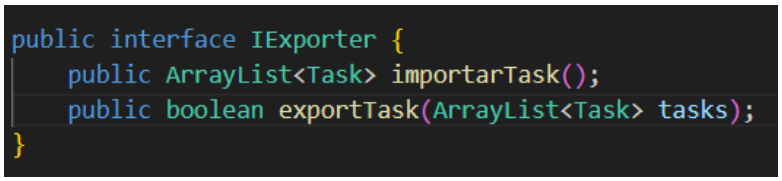
```
public boolean setExporter(String formato,String accion) {
    IExporter exporter;
    ExporterFactory factory = new ExporterFactory();
    if(formato.equals(anObject:"json")) {
        exporter= factory.getExporter(formato);
        model.setExporter(exporter);
        if(accion.equals(anObject:"e")){
            return model.exportTasksJSON();
        }else{
            return model.importTasksJSON();
        }
    }else{
        exporter= factory.getExporter(formato);
        model.setExporter(exporter);
        if(accion.equals(anObject:"e")){
            return model.exportTasksCSV();
        }else{
            return model.importTasksCSV();
        }
    }
}
```

```
public class ExporterFactory {

    public IExporter getExporter(String formato){
        IExporter exporter;
        if (formato.equals(anObject:"json")){
            exporter=new JSONExporter();
            return exporter;
        }else if(formato.equals(anObject:"csv")){
            exporter=new CSVExporter();
            return exporter;
        }else{
            return null;
        }
    }
}
```

- Acto seguido y con el exporter ya establecido se llamará al método que realice la opción seleccionada, en el modelo. Cada uno de estos cuatro métodos actúan con el repositorio de manera correcta para poder realizarse la operación deseada

Interfaz IExporter



```
public interface IExporter {
    public ArrayList<Task> importarTask();
    public boolean exportTask(ArrayList<Task> tasks);
}
```

Contiene los métodos de importación y exportación implementados tanto en la clase CSVExporter como en la clase JSONExporter. En estas dos clases se implementan de forma correcta los métodos para que funcionen.

Demostración de su Funcionamiento (CSV)

Importación

Tenemos estas 3 tareas en el archivo tasks.csv:

```
1234,Practica final, 3/12/2024,Terminar la practica final de programacion,4,100,false
4321,Examen final, 8/ 1/2025,estudiar para el examen final,3,100,true
5738,Estudiar Final, 8/ 1/2024,Estudiar para el examen final de la asignatura,3,200,false
```

Tenemos estas 2 tareas dentro del fichero serializado:

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
```

Importamos:

```
|-------------------------------------|

|---------| SUBMENU IMP/EXP |---------|

|-------------------------------------|

| 1) Importar                         |

| 2) Exportar                         |

| 3) Salir del submenu                |

|-------------------------------------|
Introduzca una opcion: 1
Introduzca el formato de importacion (csv-json)csv
Importacion exitosa
```

Resultado:

Se añade solo la tarea con identificador “4321” ya que es el único identificador diferente.

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
4321 | Examen final |  8/ 1/2025 | estudiar para el examen final | 3 | 100 | true |
```

Exportación

Tenemos estas 4 tareas:

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
949494 | Aprobar programacion III |  8/ 1/2025 | Aprobar la asignatura de programacion III | 5 | 400 | false |
3748 | Terminar la practica final | 12/12/2024 | Terminar la practica final de la asignatura | 4 | 200 | true |
```

Exportamos:

```
|-------------------------------------|

|---------| SUBMENU IMP/EXP |---------|

|-------------------------------------|

| 1) Importar                         |

| 2) Exportar                         |

| 3) Salir del submenu                |

|-------------------------------------|
Introduzca una opcion:
2
Introduzca el formato de exportacion (csv-json)csv
Exportacion exitosa
```

Resultado:

```
1234,Practica final, 3/12/2024,Terminar la practica final de programacion,4,100,false
5738,Estudiar Final, 8/ 1/2024,Estudiar para el examen final de la asignatura,3,200,false
949494,Aprobar programacion III, 8/ 1/2025,Aprobar la asignatura de programacion III,5,400,false
3748,Terminar la practica final,12/12/2024,Terminar la practica final de la asignatura,4,200,true
```

Demostración de su Funcionamiento (JSON)

Exportación

Exportamos las mismas tareas mostradas anteriormente:

```
|-------------------------------------|

|---------| SUBMENU IMP/EXP |----------|

|-------------------------------------|

| 1) Importar                         |

| 2) Exportar                         |

| 3) Salir del submenu                |

|-------------------------------------|
Introduzca una opcion: 2
Introduzca el formato de exportacion (csv-json)json
Exportacion exitosa
```

Resultado:

C: > Users > IVAN > Desktop > {} task.json > {} 3

```
[{"identifier":1234,"priority":4,"estimatedDuration":100,"title":"Practica final","content":"Terminar la practica final de programacion","date":" 3/12/2024","completed":false},
{"identifier":5738,"priority":3,"estimatedDuration":200,"title":"Estudiar Final","content":"Estudiar para el examen final de la asignatura","date":" 8/ 1/2024","completed":false},
{"identifier":949494,"priority":5,"estimatedDuration":400,"title":"Aprobar programacion III","content":"Aprobar la asignatura de programacion III","date":" 8/ 1/2025","completed":false},
{"identifier":3748,"priority":4,"estimatedDuration":200,"title":"Terminar la practica final","content":"Terminar la practica final de la asignatura","date":"12/12/2024","completed":true}]
```

Importación

Vamos a importar las siguientes tareas:

C: > Users > IVAN > Desktop > {} task.json > ...

```
[{"identifier":14993,"priority":5,"estimatedDuration":10,"title":"Demostracion JSON","content":"Demostrar la importacion de JSON","date":" 6/ 2/2024","completed":false},
{"identifier":9284,"priority":2,"estimatedDuration":120,"title":"Comprar regalos","content":"Comprar egalos para navidad","date":"28/12/2024","completed":true}]
```

Importación:

```
|-------------------------------------|

|---------| SUBMENU IMP/EXP |----------|

|-------------------------------------|

| 1) Importar                         |

| 2) Exportar                         |

| 3) Salir del submenu                |

|-------------------------------------|
Introduzca una opcion: 1
Introduzca el formato de importacion (csv-json)json
Importacion exitosa
```

Resultado:

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
949494 | Aprobar programacion III |  8/ 1/2025 | Aprobar la asignatura de programacion III | 5 | 400 | false |
3748 | Terminar la practica final | 12/12/2024 | Terminar la practica final de la asignatura | 4 | 200 | true |
14993 | Demostracion JSON |  6/ 2/2024 | Demostrar la importacion de JSON | 5 | 10 | false |
9284 | Comprar regalos | 28/12/2024 | Comprar egalos para navidad | 2 | 120 | true |
```

Funcionalidad IRepository

```java
package model;

import java.util.ArrayList;

public interface IRepository {
    public Task addTask(Task t);
    public void removeTask(Task t);
    public void modifyTask(Task t);
    public ArrayList<Task> getAllTask();

}
```

Contiene los métodos que se utilizarán en las clases que heredan de esta interfaz, en este caso BinaryRepository y NotionRepository.

Funcionalidad Repositorio Binario

Todas las demostraciones anteriores han sido realizadas dentro del repositorio binario, por lo que no se mostrarán aquí. En este apartado se mostrará el funcionamiento de la clase BinaryRepository que hereda los métodos le la interfaz IRepository.

addTask

La clase BinaryRepository contiene el ArrayList de tareas donde se almacenan las tareas.

En el método addTask comprueba si el ArrayList contiene la tarea enviada como parámetro comparando sus identificadores. En el caso de que lo contiene devuelve null y en el caso en el que no lo contiene añade la tarea y la devuelve.

```java
public class BinaryRepository implements IRepository{

    private ArrayList<Task> tasks = new ArrayList<>();

    @Override
    public Task addTask(Task t) {
        if(tasks.contains(t)){
            return null;
        }else{
            tasks.add(t);
            return t;
        }
    }
```

removeTask

Este método compara los identificadores de las tareas del ArrayList con el identificador de la tarea enviada como parámetros, si los identificadores coinciden elimina esa tarea.

```java
@Override
public void removeTask(Task t) {
    try{
        for (Task task : tasks) {
            if(task.getIdentifier()==t.getIdentifier()){
                tasks.remove(task);
            }
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

modifyTask

Este método compara los identificadores de las tareas del ArrayList hasta encontrar la tarea. Cuando la encuentra sustituye los antiguos parámetros de la tarea por los nuevos.

```java
@Override
public void modifyTask(Task t) {
    try{
        for (Task task : tasks) {
            if(task.getIdentifier()==t.getIdentifier()){
                task.setTitle(t.getTitle());
                task.setDate(t.getDate());
                task.setPriority(t.getPriority());
                task.setContent(t.getContent());
                task.setEstimatedDuration(t.getEstimatedDuration());
                task.setCompleted(t.isCompleted());
            }
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

getAllTasks

Este método devuelve el ArrayList con todas las tareas.

```java
@Override
public ArrayList<Task> getAllTask() {
    return tasks;
}
```

Funcionalidad Repositorio Notion

Cuando utilizamos el repositorio Notion también se cargan las tareas guardadas en ocasiones anteriores.

```
Página creada con ID (interno Notion)15985856-b1b2-81fb-8b27-fdad29bc4217
Datos cargados correctamente.
```

Están son las tareas cargadas:

```
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
949494 | Aprobar programacion III |  8/ 1/2025 | Aprobar la asignatura de programacion III | 5 | 400 | false |
3748 | Terminar la practica final | 12/12/2024 | Terminar la practica final de la asignatura | 4 | 200 | true |
```

Al salir del programa se guardan automáticamente:

```
Se han guardado los datos correctamente.
Saliendo...
```

Aa Identifier	Titulo	Fecha	Contenido	# Prioridad	# Duración (min)	Completado
						☐
3748	Terminar la practica final	12/12/2024	Terminar la practica final de la asignatura	4	200	☑
949494	Aprobar programacion III	8/ 1/2025	Aprobar la asignatura de programacion III	5	400	☐
5738	Estudiar Final	8/ 1/2024	Estudiar para el examen final de la asignatura	3	200	☐
1234	Practica final	3/12/2024	Terminar la practica final de programacion	4	100	☐
+ Nueva página						

Constructor NotionRepository

Este constructor recibe la Api key y el id de la base de datos como parámetro y los utiliza para crear el cliente de Notion y establecer el id de la base de datos como una cadena de caracteres. Dentro de este constructor también se configura el cliente HTTP adecuado, con los tiempos de espera y se configuran los loggers.

```java
public NotionRepository(String API_KEY, String DATABASE_ID) {
    this.client = new NotionClient(API_KEY);
    client.setHttpClient(new OkHttp5Client(60000,60000,6000));
    client.setLogger(new Slf4jLogger());
    System.setProperty(key:"notion.api.v1.logging.StdoutLogger", value:"debug");
    this.databaseId=DATABASE_ID;
}
```

addTask

Este método crea un nuevo registro en la base de datos. Para ello crea las propiedades de la página, luego configura la página padre de la base de datos, crea la solicitud a la Api de Notion y ejecuta la solicitud.

```java
@Override
public Task addTask(Task t) {
    int id = t.getIdentifier();
    String identifier = String.format(format:"%d", id);
    Map<String, PageProperty> propiedades = Map.of(
        k1:"Identifier",createTitleProperty(identifier),
        k2:"Titulo",createRichTextProperty(t.getTitle()),
        k3:"Fecha",createRichTextProperty(t.getDate()),
        k4:"Contenido",createRichTextProperty(t.getContent()),
        k5:"Prioridad",createNumberProperty(t.getPriority()),
        k6:"Duración (min)",createNumberProperty(t.getEstimatedDuration()),
        k7:"Completado",createCheckboxProperty(t.isCompleted())
    );

    PageParent parent = PageParent.database(databaseId);
    CreatePageRequest request = new CreatePageRequest(parent, propiedades);
    Page response = client.createPage(request);
    System.out.println("Página creada con ID (interno Notion)" + response.getId());
    return t;
}
```

Demostración de su funcionamiento:

```
Introduzca una opcion: 1
Introduzca el identificador unico de la tarea: 93829
Introduzca el titulo de la tarea: Demostracion Notion
Introduzca la fecha de la tarea.
Introduzca el anio: 2024
Introduzca el mes: 4
Introduzca el dia: 30
Introduzca el contenido de la tarea: Denostrar el funcionamiento de Notion
Introduzca la prioridad de la tarea(1-5)(5 maxima prioridad): 4
Introduzca la duracion estimada de la tarea en minutos: 20
Introduzca si la tarea esta completada(y/n): y
Página creada con ID (interno Notion)15985856-b1b2-815d-b824-c1284cb36ebf
Se ha añadido la tarea correctamente.
```

93829	Demostracion Notion	30/ 4/2024	Denostrar el funcionamiento de Notion	4	20	☑
3748	Terminar la practica final	12/12/2024	Terminar la practica final de la asignatura	4	200	☑
949494	Aprobar programacion III	8/ 1/2025	Aprobar la asignatura de programacion III	5	400	☐
5738	Estudiar Final	8/ 1/2024	Estudiar para el examen final de la asignatura	3	200	☐
1234	Practica final	3/12/2024	Terminar la practica final de programacion	4	100	☐

removeTask

Este método elimina un registro en función de su identificador.

```java
@Override
public void removeTask(Task t) {
    try {
        String identifier = String.format(format:"d",t.getIdentifier());
        String pageId = findPageIdByIdentifier(identifier, titleColumnName);
        if (pageId == null) {
            System.out.println("No se encontró un registro con el Identifier: " + t.getIdentifier());
            return;
        }
        UpdatePageRequest updateRequest = new UpdatePageRequest(pageId, Collections.emptyMap(), true);
        client.updatePage(updateRequest);
        System.out.println("Página archivada con ID (interno Notion)" + pageId);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Demostración de su funcionamiento:

Eliminaremos la tarea con el identificador “94572”

94572	Hacer la compra	30/11/2023	Ir a hacer la compra para esta semana	2	60	☑
93829 ABRIR	Demostracion Notion	30/ 4/2024	Denostrar el funcionan de Notion	4	20	☑
3748	Terminar la practica final	12/12/2024	Terminar la practica final de la asignatura	4	200	☑
949494	Aprobar programacion III	8/ 1/2025	Aprobar la asignatura de programacion III	5	400	☐
5738	Estudiar Final	8/ 1/2024	Estudiar para el examen final de la asignatura	3	200	☐
1234	Practica final	3/12/2024	Terminar la practica final de programacion	4	100	☐

```
Tarea eliminada correctamente.
```

Aa Identifier	Titulo	Fecha	Contenido	# Prioridad	# Duración (min)	Completado
93829	Demostracion Notion	30/ 4/2024	Denostrar el funcionamiento de Notion	4	20	☑
3748 ABRIR	Terminar la practica final	12/12/2024	Terminar la practica final de la asignatura	4	200	☑
949494	Aprobar programacion III	8/ 1/2025	Aprobar la asignatura de programacion III	5	400	☐
5738	Estudiar Final	8/ 1/2024	Estudiar para el examen final de la asignatura	3	200	☐
1234	Practica final	3/12/2024	Terminar la practica final de programacion	4	100	☐

getAllTask

Este método obtiene todos los registros y los devuelve en un ArrayList de tareas. Para ello crea una solicitud en la base de datos, ejecuta la consulta y procesa los resultados.

```java
@Override
public ArrayList<Task> getAllTask() {
    ArrayList<Task> listado = new ArrayList<>();
    try {
        QueryDatabaseRequest queryRequest = new QueryDatabaseRequest(databaseId);
        QueryResults queryResults = client.queryDatabase(queryRequest);
        for (Page page : queryResults.getResults()) {
            Map<String, PageProperty> properties = page.getProperties();
            Task task = mapPageToTask(page.getId(),properties);
            if (task != null) {
                listado.add(task);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return listado;
}
```

Demostración de su funcionamiento:

Aa Identifier	Titulo	Fecha	Contenido	# Prioridad	# Duración (min)	Completado
94572	Hacer la compra	30/11/2023	Ir a hacer la compra para esta semana	2	60	☑
93829	Demostracion Notion	30/ 4/2024	Denostrar el funcionamiento de Notion	4	20	☑
3748	Terminar la practica final	12/12/2024	Terminar la practica final de la asignatura	4	200	☑
949494	Aprobar programacion III	8/ 1/2025	Aprobar la asignatura de programacion III	5	400	☐
5738	Estudiar Final	8/ 1/2024	Estudiar para el examen final de la asignatura	3	200	☐
1234	Practica final	3/12/2024	Terminar la practica final de programacion	4	100	☐

Y si pedimos que nos muestre las tareas.

```
94572 | Hacer la compra | 30/11/2023 | Ir a hacer la compra para esta semana | 2 | 60 | true |
93829 | Demostracion Notion | 30/ 4/2024 | Denostrar el funcionamiento de Notion | 4 | 20 | true |
3748 | Terminar la practica final | 12/12/2024 | Terminar la practica final de la asignatura | 4 | 200 | true |
949494 | Aprobar programacion III |  8/ 1/2025 | Aprobar la asignatura de programacion III | 5 | 400 | false |
5738 | Estudiar Final |  8/ 1/2024 | Estudiar para el examen final de la asignatura | 3 | 200 | false |
1234 | Practica final |  3/12/2024 | Terminar la practica final de programacion | 4 | 100 | false |
```

ModifyTask

Este método modifica la tarea seleccionada por su identificador. Para ello busca la tarea, crea las propiedades actualizada y crea la solicitud de actualización.

```java
@Override
public void modifyTask(Task t) {
    try {
        String identifier = String.format(format:"d",t.getIdentifier());
        String pageId = findPageIdByIdentifier(identifier,titleColumnName);
        if (pageId == null) {
            System.out.println("No se encontró un registro con el Identifier: " + t.getIdentifier());
            return;
        }
        Map<String, PageProperty> updatedProperties = Map.of(
            k1:"Titulo",createRichTextProperty(t.getTitle()),
            k2:"Fecha",createRichTextProperty(t.getDate()),
            k3:"Contenido",createRichTextProperty(t.getContent()),
            k4:"Prioridad",createNumberProperty(t.getPriority()),
            k5:"Duración (min)",createNumberProperty(t.getEstimatedDuration()),
            k6:"Completado",createCheckboxProperty(t.isCompleted())
        );
        UpdatePageRequest updateRequest = new UpdatePageRequest(pageId, updatedProperties);
        client.updatePage(updateRequest);

        System.out.println("Página actualizada con ID (interno Notion)" + pageId);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Demostración de su funcionamiento:

```
Introduzca el identificador de la tarea que desee modificar: 1234
Introduzca el nuevo titulo de la tarea: Demostrar la modificacion
Introduzca la fecha de la tarea.
Introduzca el anio: 2022
Introduzca el mes: 12
Introduzca el dia: 3
Introduzca el nuevo contenido de la tarea: Demostrar que se ha modificado la informacion de manera correcta
Introduzca la nueva prioridad de la tarea(1-5)(5 maxima prioridad): 4
Introduzca la nueva duracion estimada de la tarea en minutos: 30
Introduzca si la tarea esta completada(y/n): n
```

1234	Demostrar la modificacion	3/12/2022	Demostrar que se ha modificado la informacion de manera correcta	4	30	☐

Funcionalidad final

Una vez el usuario quiere finalizar la ejecución el código entra en el método del controlador “end”. Verifica que se han guardado todos los datos correctamente y manda un mensaje de despedida.

```java
public void end(){
    if(model.saveData()){
        view.showMessage(msg:"Se han guardado los datos correctamente.");
    }else{
        view.showErrorMessage(msg:"No se ha podido guardar el nuevo estado.");
    }
    view.end();
}
```

El método saveData se encarga de guardar todas las tareas en un fichero binario para que la próxima vez que se ejecute el código poder recuperar la información.

```java
public boolean saveData(){
    ObjectOutputStream oos = null;
    try {
        oos = new ObjectOutputStream(new FileOutputStream(ficheroSerializado));
        ArrayList<Task> listado = repository.getAllTask();
        oos.writeObject(listado);
        return true;
    } catch (IOException ex) {
        System.err.println("Error durante la serialización: " + ex.getMessage());
        return false;
    } finally {
        if (oos != null) {
            try {
                oos.close();
            } catch (IOException ex) {
                System.err.println("Error al cerrar el flujo: " + ex.getMessage());
                return false;
            }
        }
    }
}
```

```
Se han guardado los datos correctamente.
Saliendo...
```