

Universidad de las Fuerzas Armadas - ESPE
MODALIDAD EN LÍNEA



Informe de Proyecto Segundo Parcial.

Curso: Programación Orientada a Objetos

Profesor: Ing. Luis Enrique Jaramillo Montaña.

Fecha: 06/01/2025

1. Título del Proyecto:

Aplicación de Gestión de Tareas con Interfaz
Gráfica y Base de Datos NoSQL

2. Datos de los estudiantes:

- Jefferson Fernando Jaya Taipe.
- Isaac Sebastian Proaño Panchi
- Nick Mateo Ortiz Córdova
- Nayeli Estefania Vilaña Puga.

3. NRC:1322

4. Fecha de Entrega:

- 14/02/2025

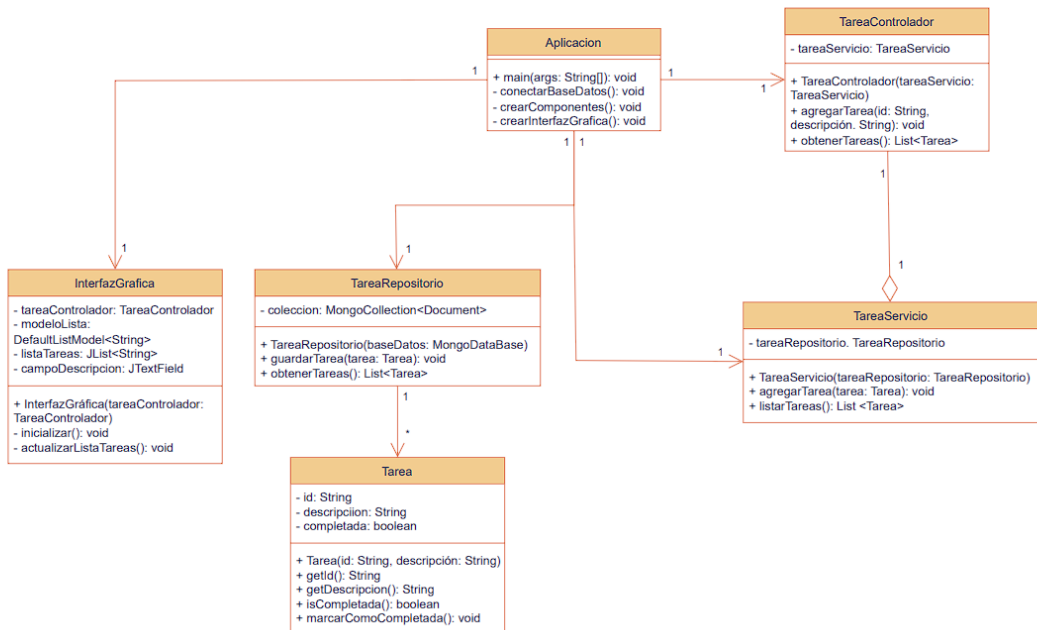
Objetivo de la actividad:

El propósito central de esta tarea es desarrollar e instaurar una aplicación de administración de tareas que fusione una interfaz de usuario visualmente intuitiva con una base de datos NoSQL eficaz. Además, se pretende implementar los fundamentos SOLID en el diseño.

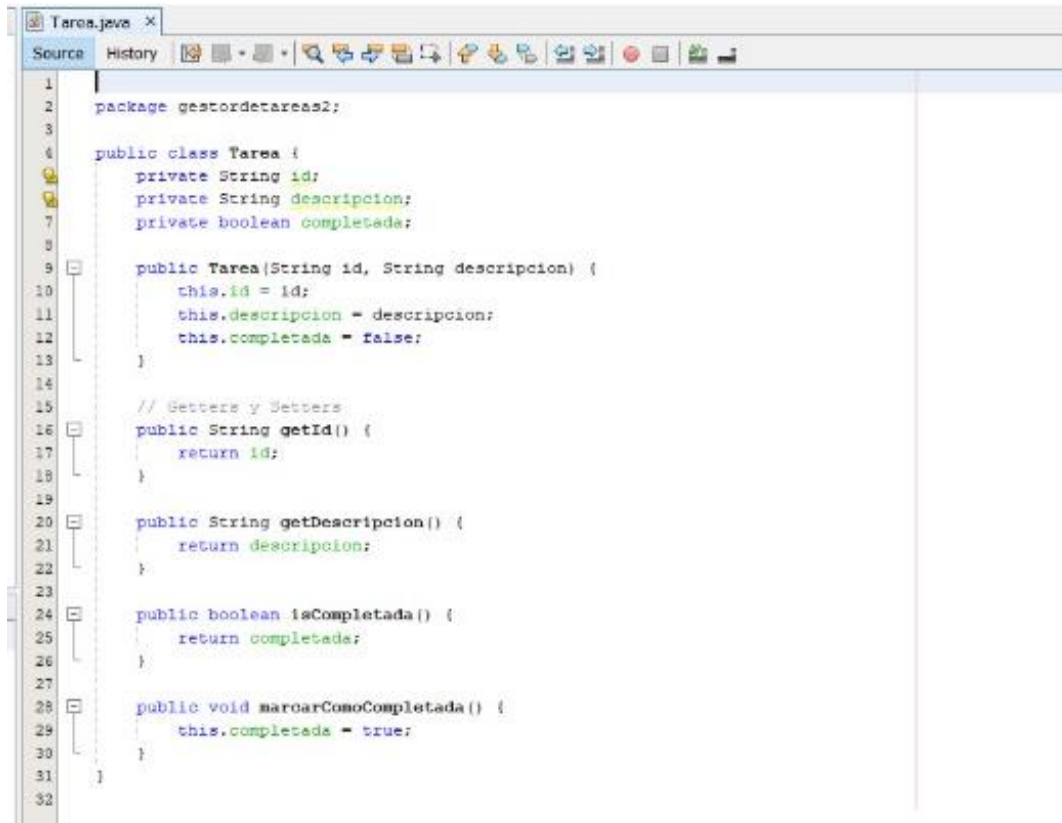
Descripción de la Actividad:

Esta tarea implica la elaboración de una aplicación de administración de tareas que habilite a los usuarios para añadir, modificar, suprimir y mostrar tareas de forma eficaz. Es fundamental que la interfaz gráfica sea intuitiva y promueva la interacción del usuario con la aplicación. Para el almacenamiento de información, se empleará una base de datos NoSQL, escogida por su versatilidad y capacidad de expansión. En el proceso de desarrollo, se pondrán en práctica los principios SOLID para garantizar un código puro.

Desarrollo de la Actividad:**1. Diagrama UML**



2. Código

A screenshot of an IDE window titled 'Tarea.java'. The window has a 'Source' tab and a toolbar with various icons. The code is written in Java and defines a 'Tarea' class. The code is as follows:

```
1
2 package gestordetareas2;
3
4 public class Tarea {
5     private String id;
6     private String descripcion;
7     private boolean completada;
8
9     public Tarea(String id, String descripcion) {
10         this.id = id;
11         this.descripcion = descripcion;
12         this.completada = false;
13     }
14
15     // Getters y Setters
16     public String getId() {
17         return id;
18     }
19
20     public String getDescripcion() {
21         return descripcion;
22     }
23
24     public boolean isCompletada() {
25         return completada;
26     }
27
28     public void marcarComoCompletada() {
29         this.completada = true;
30     }
31 }
32
```

```
TareaControlador.java x
Source History
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package gestordetareas2;
7
8   import java.util.List;
9
10  public class TareaControlador {
11      private final TareaServicio tareaServicio;
12
13      public TareaControlador(TareaServicio tareaServicio) {
14          this.tareaServicio = tareaServicio;
15      }
16
17      public void agregarTarea(String id, String descripcion) {
18          Tarea tarea = new Tarea(id, descripcion);
19          tareaServicio.agregarTarea(tarea);
20      }
21
22      public List<Tarea> obtenerTareas() {
23          return tareaServicio.listarTareas();
24      }
25  }
```

```
TareaServicio.java x
Source History
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package gestordetareas2;
7
8   import java.util.List;
9
10  public class TareaServicio {
11      private final TareaRepositorio tareaRepositorio;
12
13      public TareaServicio(TareaRepositorio tareaRepositorio) {
14          this.tareaRepositorio = tareaRepositorio;
15      }
16
17      public void agregarTarea(Tarea tarea) {
18          tareaRepositorio.guardarTarea(tarea);
19      }
20
21      public List<Tarea> listarTareas() {
22          return tareaRepositorio.obtenerTareas();
23      }
24  }
```

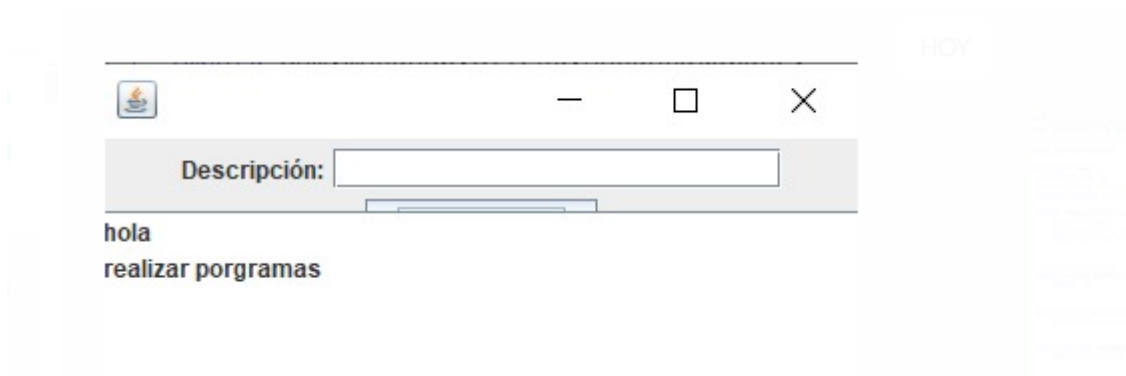
```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package gestordetareas3;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12 import java.util.List;
13
14 public class InterfazGrafica extends JFrame {
15     private final TareaControlador tareaControlador;
16     private final DefaultListModel<String> modeloLista;
17     private final JList<String> listaTareas;
18     private final JTextField campoDescripcion;
19
20     public InterfazGrafica(TareaControlador tareaControlador) {
21         this.tareaControlador = tareaControlador;
22         this.modeloLista = new DefaultListModel<>();
23         this.listaTareas = new JList<>(modeloLista);
24         this.campoDescripcion = new JTextField(20);
25         inicializar();
26     }
27
28     private void inicializar() {
29         setLayout(new BorderLayout());
30
31         // Panel para agregar tareas
32         JPanel panelAgregar = new JPanel();
33         panelAgregar.add(new JLabel("Descripción:"));
34         panelAgregar.add(campoDescripcion);
35         JButton botonAgregar = new JButton("Agregar Tarea");
36         panelAgregar.add(botonAgregar);
37         add(panelAgregar, BorderLayout.NORTH);
38
39         // Lista de tareas
40         add(new JScrollPane(listaTareas), BorderLayout.CENTER);
41
42         // Acción del botón agregar
43         botonAgregar.addActionListener(new ActionListener() {
44             @Override
45             public void actionPerformed(ActionEvent e) {
46                 String descripcion = campoDescripcion.getText();
47                 if (!descripcion.isEmpty()) {
48                     String id = String.valueOf(System.currentTimeMillis()); // Generar un ID único
49                     tareaControlador.agregarTarea(id, descripcion);
50                     actualizarListaTareas();
51                     campoDescripcion.setText("");
52                 } else {
53                     JOptionPane.showMessageDialog(InterfazGrafica.this, "La descripción no puede estar vacía.", "Error", JOptionPane.ERROR_MESSAGE);
54                 }
55             }
56         });
57
58         // Cargar tareas al iniciar
59         actualizarListaTareas();
60     }
61
62     private void actualizarListaTareas() {
63         modeloLista.clear();
64         List<Tarea> tareas = tareaControlador.obtenerTareas();
65         for (Tarea tarea : tareas) {
66             modeloLista.addElement(tarea.getDescripcion() + (tarea.isCompletada() ? " (Completada)" : ""));
67         }
68     }
69 }

```

```
Aplicacion.java x
Source History
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package gestordetareas2;
7
8   import com.mongodb.MongoClient;
9   import com.mongodb.client.MongoDatabase;
10
11  import javax.swing.*;
12
13  public class Aplicacion {
14      public static void main(String[] args) {
15          // Conectar a la base de datos MongoDB
16          MongoClient mongoClient = new MongoClient("localhost", 27017);
17          MongoDatabase baseDatos = mongoClient.getDatabase("gestor_tareas");
18
19          // Crear el repositorio, servicio y controlador
20          TareaRepositorio tareaRepositorio = new TareaRepositorio(baseDatos);
21          TareaServicio tareaServicio = new TareaServicio(tareaRepositorio);
22          TareaControlador tareaControlador = new TareaControlador(tareaServicio);
23
24          // Crear la interfaz gráfica
25          InterfazGrafica interfazGrafica = new InterfazGrafica(tareaControlador);
26          interfazGrafica.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27          interfazGrafica.setSize(400, 300);
28          interfazGrafica.setVisible(true);
29      }
30  }
31
```

Ejecución del código



Resumen de Resolución:

En la actividad se realizó un estudio de los requerimientos funcionales y no funcionales de la aplicación. Se eligió una base de datos NoSQL apropiada para gestionar la estructura de datos necesaria y se desarrolló una interfaz visual enfocada en la experiencia del usuario. La estructuración de la arquitectura del software se basó en los principios SOLID, lo que posibilitó una distinción nítida de responsabilidades y simplificó futuras ampliaciones o cambios del sistema. Se llevaron a cabo ensayos rigurosos para asegurar el funcionamiento adecuado de todas las funcionalidades incorporadas.

Conclusión:

La puesta en marcha de una aplicación de administración de tareas con una interfaz gráfica y una base de datos NoSQL, basada en los principios SOLID, culminó en un sistema eficaz y de fácil mantenimiento. La aplicación alcanza las metas establecidas, proporcionando a los usuarios un instrumento eficaz para la administración de sus actividades cotidianas. La implementación de los principios SOLID se muestra ventajosa, ofreciendo una base firme para el avance y la expansión futura del software.

Referencias:

Romero, A. C., Sanabria, J. S. G., & Cuervo, M. C. (2012). Utilidad y funcionamiento de las bases de datos NoSQL. *Facultad de Ingeniería*, 21(33), 21-32.

Alloza Álvarez, E. M. (2015). *Aplicación de Android para geo-localización de tareas pendientes. Implementación y gestión de servidor, comunicación e interfaz gráfica* (Doctoral dissertation, Universitat Politècnica de València).

del Busto, H. G., & Enríquez, O. Y. (2013). Bases de datos NoSQL. *Telemática*, 11(3),

21-33.

Sánchez, G. (2023). Principios SOLID en la Automatización de Pruebas de Software para Interfaces de Usuario Web con Selenium WebDriver y Java. *Revista Politécnica de Aguascalientes*, 2.

