

Universidad de Guayaquil

Facultad de Ciencias Matemáticas y Físicas

Materia:

Verificación y Validación.

PROYECTO

Integrantes:

- Asitimbay Shigla Stalyn
- Melendres Sánchez Nallely
- Monge Yasbek Frank Richard
- Guillén Salabarría Jonatán Josué
- Riera Rizzo Keneth Bryan
- Bailón Guaranda Marcos

Ciclo:

2023-2024 CI

Contenido

Desarrollo	1
1. Introducción al Análisis Estático y Dinámico.....	1
2. Importancia	2
3. Objetivos	3
3.1. Objetivo general.....	3
3.2. Objetivos específicos.....	3
4. Técnicas y Herramientas	4
5. Importancia del análisis de complejidad y aplicación de estándares	8
6. Aplicación de Técnicas	9
7. Aplicación de Herramientas	14
8. Estándares y Guías en el Desarrollo.....	16
• Java Code Conventions	16
• Java de Google	17
9. Aplicación de V&V en un Programa de Administración de una Escuela	18
10. Requerimientos.....	20
10.1. Introducción.....	20
10.2. Descripción general.....	21
10.3. Requisitos de la interfaz externa	23
10.4. Modelo de dominio	25
10.5. System Features (Casos de uso).....	25
10.6. Requisitos Funcionales.....	37
10.7. Requerimientos No Funcionales	37
10.8. Otros requisitos no funcionales	38
10.9. Otros requerimientos.....	39
Apéndice A: Glosario.....	39
Apéndice B: Modelo de Análisis.....	39
Apéndice C: To Be Determined List.....	40
11. Manual Técnico	40
11.1. Objetivos.....	40
11.2. Objetivos Específicos.....	40

11.3.	Alcance.....	41
11.4.	Requerimientos Mínimos de Hardware.....	41
11.5.	Requerimientos Mínimos de Software.....	41
11.6.	Herramientas Utilizadas para el Desarrollo	41
11.7.	Instalación	42
11.8.	Guía de Uso.....	46
11.9.	Usuarios de sistema operativo	55
11.10.	Contingencias y soluciones	55
	Conclusiones	57
	Referencias Bibliográficas	58

Desarrollo

1. Introducción al Análisis Estático y Dinámico.

El análisis estático y dinámico son dos enfoques complementarios utilizados en el campo de la programación y la ingeniería de software para evaluar y mejorar la calidad del código. Ambos métodos tienen como objetivo identificar posibles errores, vulnerabilidades y mejorar el rendimiento de las aplicaciones, pero difieren en su enfoque y momento de aplicación. El análisis estático se realiza sin ejecutar el código y se centra en examinar el código fuente o el artefacto binario para detectar problemas potenciales. El análisis estático es altamente automatizado y utiliza técnicas como el análisis léxico, sintáctico y semántico para revisar la estructura, la coherencia y la lógica del código. Algunas herramientas populares de análisis estático incluyen analizadores estáticos de código y herramientas de revisión de estilo. El análisis estático ayuda a identificar problemas comunes como errores de sintaxis, uso incorrecto de variables, vulnerabilidades de seguridad, fugas de memoria y malas prácticas de programación. Al realizar un análisis estático, los desarrolladores pueden encontrar y corregir problemas potenciales antes de que se ejecuten en un entorno de producción, lo que ayuda a mejorar la calidad y la fiabilidad del software.

El análisis dinámico, por otro lado, se realiza durante la ejecución del programa. Implica probar el software utilizando diferentes casos de prueba y escenarios para observar su comportamiento en tiempo real. Las técnicas de análisis dinámico incluyen pruebas de unidad, pruebas de integración, pruebas de rendimiento. El análisis dinámico permite evaluar cómo se comporta el software en condiciones reales, identificar posibles errores, cuellos de botella de rendimiento y problemas de seguridad que pueden surgir solo durante la ejecución. Esta forma de análisis es crucial para garantizar que el software funcione correctamente y cumpla con los requisitos establecidos. Permite descubrir errores que pueden haber pasado desapercibidos durante el análisis estático y proporciona información valiosa para mejorar y optimizar el rendimiento de la aplicación.

2. Importancia.

La importancia del análisis estático y dinámico radica en su capacidad para mejorar la calidad del software y garantizar su correcto funcionamiento.

Importancia del análisis estático:

1. Identificación temprana de errores: El análisis estático permite detectar errores de sintaxis, inconsistencias y malas prácticas en el código fuente antes de la ejecución del programa. Esto ayuda a corregir los errores en etapas tempranas del desarrollo, evitando problemas posteriores y reduciendo los costos asociados con la corrección de errores.
2. Mejora de la calidad del código: Mediante el análisis estático, se pueden identificar patrones y prácticas de programación que no cumplen con los estándares establecidos. Esto ayuda a mantener un código más limpio, legible y coherente, lo que facilita su mantenimiento y comprensión por parte de otros desarrolladores.
3. Detección de vulnerabilidades de seguridad: El análisis estático puede revelar posibles vulnerabilidades en el código, como la falta de validación de entradas, la exposición de datos sensibles o la utilización incorrecta de funciones criptográficas. Identificar y corregir estas vulnerabilidades antes de la implementación del software es esencial para proteger la integridad y la seguridad de la aplicación.
4. Optimización del rendimiento: El análisis estático puede ayudar a identificar áreas del código que pueden afectar negativamente el rendimiento de la aplicación, como bucles innecesarios, uso ineficiente de recursos o algoritmos ineficientes. Optimizar el código en base a los hallazgos del análisis estático puede resultar en un mejor rendimiento y una mayor eficiencia del software.

Importancia del análisis dinámico:

1. Validación del comportamiento en tiempo real: El análisis dinámico permite observar el comportamiento real del software durante su ejecución. Esto ayuda a verificar que la aplicación se comporte según lo esperado, cumpla con los requisitos funcionales y ofrezca una experiencia de usuario satisfactoria.

2. Detección de errores y excepciones: Durante la ejecución del programa, el análisis dinámico puede identificar errores, excepciones y condiciones inesperadas que no se pueden descubrir mediante el análisis estático. Esto permite corregir problemas que solo se manifiestan en tiempo de ejecución y garantizar la estabilidad y la robustez del software.
3. Evaluación del rendimiento y la escalabilidad: El análisis dinámico puede medir y evaluar el rendimiento del software en condiciones reales, identificando posibles cuellos de botella, tiempos de respuesta lentos o problemas de escalabilidad. Esto ayuda a optimizar el rendimiento y asegurar que el software pueda manejar adecuadamente las cargas de trabajo previstas.
4. Verificación de requisitos no funcionales: El análisis dinámico puede ayudar a verificar requisitos no funcionales, como la seguridad, la usabilidad o la accesibilidad. Mediante pruebas exhaustivas y escenarios de uso real, se puede evaluar si el software cumple con estos requisitos y tomar medidas correctivas si es necesario.

3. Objetivos

3.1. Objetivo general:

- Comprender y aplicar las técnicas de análisis estático y dinámico en el proceso de Verificación y Validación (V&V) del software.

3.2. Objetivos específicos:

- Familiarizarse con los conceptos fundamentales del análisis estático y dinámico.
- Identificar y analizar las herramientas utilizadas en el análisis estático y dinámico, como JUnit, Checkstyle y CyVis.
- Aplicar técnicas de análisis estático para evaluar la estructura, coherencia y lógica del código fuente.
- Evaluar la complejidad del código utilizando técnicas como la complejidad ciclomática.
- Aplicar estándares y guías de desarrollo, Java de Google, para mejorar la calidad y legibilidad del código.

- Aplicar técnicas de Verificación y Validación (V&V) en un programa de administración de una escuela como caso de estudio.
- Evaluar la efectividad de las técnicas de V&V aplicadas en el proyecto de administración de una escuela.

4. Técnicas y Herramientas.

En nuestro proceso de Verificación y Validación (V&V) del software, utilizamos varias técnicas para garantizar la calidad y confiabilidad del producto final. Una de las técnicas fundamentales que aplicamos es el análisis estático, con un enfoque particular en el análisis de complejidad. El análisis estático nos permite examinar el código fuente sin ejecutarlo, con el objetivo de identificar posibles problemas y riesgos antes de que se produzcan errores durante la ejecución. En este sentido, utilizamos el análisis de complejidad como una técnica específica dentro del análisis estático.

El análisis de complejidad nos permite evaluar la estructura y el diseño del código fuente para determinar su nivel de complejidad. Al analizar la complejidad, buscamos patrones y características que puedan dificultar la comprensión, mantenibilidad y escalabilidad del software. Algunos ejemplos de aspectos que evaluamos incluyen la longitud y anidación excesiva de los métodos, el acoplamiento excesivo entre componentes, la falta de modularidad y la duplicación de código.

Para respaldar nuestro análisis de complejidad, utilizamos la herramienta Checkstyle. Es una herramienta de análisis estático de código para Java que nos ayuda a verificar el cumplimiento de las guías de codificación establecidas por Google. Estas guías representan un conjunto de mejores prácticas y estándares de codificación que promueven la legibilidad, consistencia y calidad del código. Checkstyle examina el código fuente y verifica su conformidad con las reglas y estándares definidos en las guías de Google. Algunos aspectos que evalúa incluyen el estilo de nomenclatura, el formato del código, el manejo adecuado de excepciones y la complejidad de los métodos. Si se encuentran violaciones a las reglas establecidas, la herramienta genera informes que nos permiten identificar y corregir rápidamente dichas violaciones. La combinación del análisis de complejidad y el uso de

Checkstyle nos brinda una visión más completa y precisa de la calidad del código fuente. Al evaluar la complejidad, podemos identificar áreas problemáticas que podrían llevar a errores y dificultades en el mantenimiento del software. Al mismo tiempo, el uso de Checkstyle nos ayuda a garantizar que el código cumpla con las directrices y estándares establecidos, lo que contribuye a la legibilidad y consistencia del código.

Además del análisis de complejidad y Checkstyle, durante las actividades impartidas en clase utilizamos otras herramientas que complementaron nuestro proceso de V&V. Estas incluyeron JUnit, una herramienta de pruebas unitarias para evaluar la funcionalidad de componentes individuales del software, y CyVis, una herramienta de análisis estático de código adicional que nos ayudó a detectar problemas de calidad y buenas prácticas de programación.

En el siguiente cuadro comparativo, presentamos las principales características de cada una de las herramientas utilizadas durante el parcial, destacando su utilidad y contribución al proceso de V&V.

Análisis Comparativo de las Herramientas.

Aspecto	Checkstyle	CyVis	JUnit
Propósito	Herramienta de análisis estático de código para detectar violaciones de estilo de codificación.	Herramienta de visualización de dependencias y métricas de código fuente.	Marco de prueba unitaria para el desarrollo de software en Java.
Funcionalidad	Verifica el cumplimiento de convenciones de estilo de codificación, identifica y resalta problemas de codificación, y	Genera visualizaciones de dependencias, permite identificar estructuras de código complejas y	Proporciona un entorno de prueba para escribir y ejecutar casos de prueba unitarios, facilitando la detección temprana de

	proporciona recomendaciones para mejorar la calidad del código.	proporciona métricas de código para ayudar en la gestión del proyecto.	errores y problemas en el código.
Integración	Puede integrarse con IDEs como Eclipse, IntelliJ IDEA y NetBeans, así como con sistemas de construcción como Maven y Ant.	No se integra directamente con IDEs o sistemas de construcción.	Puede integrarse con IDEs populares como Eclipse, IntelliJ IDEA y NetBeans, así como con sistemas de construcción como Maven y Gradle.
Reglas de estilo	Proporciona una amplia gama de reglas de estilo personalizables para verificar el código fuente.	No se centra en la aplicación de reglas de estilo, ya que su enfoque principal es la visualización y las métricas del código fuente.	No se centra en la aplicación de reglas de estilo, ya que su objetivo principal es facilitar las pruebas unitarias.
Salida de informe	Genera informes detallados que indican las violaciones de estilo encontradas y proporcionan recomendaciones para corregirlas.	Genera visualizaciones gráficas de dependencias y métricas de código fuente en formatos como gráficos de llamadas, diagramas de paquetes, entre otros.	Proporciona informes de resultados de pruebas unitarias, mostrando si las pruebas han sido exitosas o fallidas.
Aplicación	Es útil durante el proceso de desarrollo para mantener un código limpio y	Es útil para analizar la complejidad y las dependencias del código fuente, lo	Es fundamental para escribir y ejecutar pruebas unitarias, lo que facilita la detección temprana de

	consistente, evitando problemas comunes de codificación.	que puede ayudar en la planificación y toma de decisiones del proyecto.	errores y asegura la calidad del código.
Lenguaje	Es compatible con múltiples lenguajes de programación, incluyendo Java, C/C++, C#, Python, entre otros.	Es compatible principalmente con Java, aunque también tiene soporte limitado para otros lenguajes como C#.	Es específico para pruebas unitarias en Java, aunque también tiene soporte para otros lenguajes a través de bibliotecas adicionales.
Popularidad	Es ampliamente utilizado en la comunidad de desarrollo de software y cuenta con una gran base de usuarios.	Tiene menos popularidad en comparación con Checkstyle y JUnit, pero es utilizado por desarrolladores interesados en visualizar dependencias y métricas del código fuente.	Es uno de los marcos de prueba unitaria más populares para Java y es ampliamente utilizado en la comunidad de desarrollo de software.

Finalmente, el uso de herramientas como Checkstyle, CyVis y JUnit durante el desarrollo de software puede brindar beneficios significativos. El estilo de verificación le permite mantener un código limpio y coherente al verificar el cumplimiento de las convenciones de estilo de codificación. CyVis proporciona visualización de código fuente y métricas para ayudar en la planificación de proyectos y la toma de decisiones. Finalmente, JUnit es un marco básico de prueba de unidades que lo ayuda a encontrar errores temprano y garantizar la calidad del código. Aunque estas herramientas sirven para diferentes propósitos, juntas ayudan a mejorar la calidad, la confiabilidad y la capacidad de mantenimiento del software. Su popularidad y uso generalizado en la comunidad de desarrollo de software respalda su validez y valor en la industria.

5. Importancia del análisis de complejidad y aplicación de estándares.

El análisis de complejidad y la aplicación de estándares son dos aspectos interrelacionados y complementarios del desarrollo de software, que son esenciales para la creación de sistemas de calidad.

- **Mejore el diseño y la calidad del código:** el análisis de complejidad ayuda a identificar áreas de código muy complejo que pueden indicar problemas de diseño o incumplimiento de los estándares de codificación. Mediante el uso de estándares de codificación coherentes, el código se simplifica y se lee, lo que reduce la complejidad y facilita su comprensión y mantenimiento.
- **Eficiencia y rendimiento del software:** el análisis de complejidad le permite evaluar el impacto de la complejidad en el rendimiento del software. Aplicando criterios de optimización y buenas prácticas de programación, puedes eliminar cuellos de botella y mejorar la eficiencia del código evitando operaciones innecesarias o ineficientes. Esto ayuda a mejorar el rendimiento del software y proporciona una experiencia más fluida para los usuarios.
- **Mantenimiento y desarrollo de software:** una combinación de análisis de complejidad y aplicación de estándares facilita el mantenimiento y desarrollo de software a lo largo del tiempo. Identificar áreas de código complejas y aplicar estándares de codificación claros simplifica la comprensión del código y reduce la dependencia de la experiencia específica del desarrollador original. Esto facilita agregar nuevos miembros al equipo de desarrollo y adaptar y ampliar el software a medida que cambian los requisitos.
- **Cumplimiento de regulaciones y estándares de seguridad:** una combinación de análisis de complejidad y la aplicación de estándares de seguridad asegura que el desarrollo de software cumpla con regulaciones y estándares específicos. La evaluación de la complejidad del código y el uso de estándares de seguridad reconocidos pueden reducir el riesgo de vulnerabilidades y garantizar la protección de datos y la integridad del sistema.

El análisis de complejidad y la aplicación de estándares juntos contribuyen a un desarrollo de software más estable, eficiente y mantenible.

Al simplificar el código, mejorar la calidad del diseño y garantizar el cumplimiento de los estándares, puede crear sistemas de alta calidad que cumplan con los requisitos funcionales y no funcionales y que sean más fáciles de mantener, evolucionar y proteger con el tiempo.

6. Aplicación de Técnicas.

Aplicamos la técnica de análisis de complejidad para evaluar la estructura y el nivel de complejidad de nuestro software. Esta técnica nos permite identificar áreas problemáticas, detectar patrones de complejidad y evaluar métricas clave relacionadas con el código. Para llevar a cabo el análisis de complejidad, seguimos los siguientes pasos:

1. Revisión de la estructura del código: Analizamos la organización y la arquitectura del código para comprender su complejidad global y la interacción entre los diferentes componentes.
2. Identificación de patrones de complejidad: Buscamos patrones comunes asociados con la complejidad, como bucles anidados, excesiva dependencia entre clases o métodos largos y complejos.
3. Evaluación de métricas de complejidad: Utilizamos métricas como el número de líneas de código, la complejidad ciclomática y el acoplamiento entre clases para cuantificar la complejidad del software y establecer umbrales de aceptación.

Al aplicar esta técnica de análisis de complejidad, podemos identificar áreas de mejora y tomar acciones correctivas para reducir la complejidad y mejorar la mantenibilidad del software. A continuación, presentamos los grafos generados a partir de estos análisis, los cuales nos permiten visualizar de manera clara y concisa la complejidad de nuestro software:

EJERCICIOS DE COMPLEJIDAD

Caso de uso 1: Ingresar a la cuenta

Sentencias	Grafo	Complejidad
<pre> boolean registroExitoso = registrarUsuario(String usuario, String contraseña); if (registroExitoso) { System.out.println("Registro exitoso"); } else { } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$

Caso de uso 2: Creación de Contraseñas Seguras

Sentencias	Grafo	Complejidad
<pre> if (verificarContraseña(contraseña)) { System.out.println("Contraseña registrada exitosamente. ¡Es segura!"); } else { System.out.println("La contraseña no cumple con los requisitos de seguridad."); } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$

Caso de uso 3: Ingresar a Cuenta

Sentencias	Grafo	Complejidad
<pre> boolean iniciarSesion(String usuario, String contraseña){ if(!usuario.isEmpty() && !contraseña.isEmpty()){ if(usuario=="admin" && contraseña=="admin"){ System.out.println("Ingreso de cuenta exitoso"); }else{ System.out.println("Datos incorrectos"); } }else{ System.out.println("Llene todo los campos") } } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 4 + 1$ $M = 5$ $R = 5$

Caso de uso 4: Gestionar Cuentas		
Sentencias	Grafo	Complejidad
<pre> Switch (opcion){ 1 case 1: Agregar(); break; 2 case 2: Editar(); break; 3 case 3: Visualizar(); break; 4 case 4: Eliminar(); break; 5 default: break; } </pre>	<pre> graph TD I((I)) --> 1((1)) I --> 2((2)) I --> 3((3)) I --> 4((4)) I --> 5((5)) 1 --> F((F)) 2 --> F 3 --> F 4 --> F 5 --> F </pre>	$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 5 + 1$ $M = 6$ $R = 6$

Caso de uso 5: Filtrar por edad y género.		
Sentencias	Grafo	Complejidad
<pre> boolean filtrar(int edad, String genero){ 1 if(!edad.isEmpty() && !genero.getSelectedItem()){ 2 3 //Mostrar datos que coincidan con parámetros }else{ 4 System.out.println("Campos vacíos") } } </pre>	<pre> graph TD I((I)) --> 1((1)) 1 --> 2((2)) 2 --> 3((3)) 3 --> 4((4)) 4 --> F((F)) </pre>	$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 2 + 1$ $M = 3$ $R = 3$

Caso de uso 6: Guardar los datos en el dispositivo.		
Sentencias	Grafo	Complejidad
<pre> guardar = cuadro.getSelectedFile(); if (guardar != null) { 1 escribir = new FileWriter(guardar + ".mabel", false); 2 for (Docente d : docentes) { 3 4 escribir.write(d.getCedula() + "\n"); 5 escribir.write(d.getNombre() + "\n"); 6 escribir.write(d.getSexo() + "\n"); 7 escribir.write(d.getFechaNac().toString() + "\n"); 8 escribir.write(d.getTelefono() + "\n"); 9 escribir.write(d.getDireccion() + "\n"); 10 escribir.write(d.getFacultad() + "\n"); 11 escribir.write(d.getTitulo() + "\n"); 12 escribir.write(d.getMateria() + "\n"); } 13 escribir.close(); } </pre>	<pre> graph TD I((I)) --> 1((1)) 1 --> 2((2)) 2 --> 3((3)) 3 --> 4((4)) 4 --> 5((5)) 5 --> 6((6)) 6 --> 7((7)) 7 --> 8((8)) 8 --> 9((9)) 9 --> 10((10)) 10 --> 11((11)) 11 --> 12((12)) 12 --> 13((13)) 13 --> F((F)) </pre>	$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 0 + 1$ $M = 1$ $R = 1$

Caso de uso 7: Gestión de roles de permiso.		
Sentencias	Grafo	Complejidad
<pre> public static void asignarRolesPermisos() { // Obtener la lista de usuarios a los que se les asignarán los roles // y permisos 1 List<Usuario> usuarios = obtenerUsuarios(); // Mostrar una interfaz para seleccionar los roles y permisos 2 mostrarInterfazRolesPermisos(); // Asignar los roles y permisos seleccionados a cada usuario for (Usuario usuario : usuarios) { 3 asignarRolesPermisosUsuario(usuario); } 4 } </pre>	<pre> graph TD I((I)) --> 1((1)) 1 --> 2((2)) 2 --> 3((3)) 3 --> 4((4)) 4 --> F((F)) </pre>	$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 0 + 1$ $M = 1$ $R = 1$

Caso de uso 8: Gestion de archivos.		
Sentencias	Grafo	Complejidad
<pre> if (resp == JFileChooser.APPROVE_OPTION) { datos = new DatosDocente(); contenido = datos.guardarArchivo(docentes, cuadro.getSelectedFile()); } else if (contenido.charAt(0) == '1') { JOptionPane.showMessageDialog(null, "DATOS GUARDADOS EXITOSAMENTE"); } else if (contenido.charAt(0) == '0') { JOptionPane.showMessageDialog(null, "ERROR AL GUARDAR LA INFORMACIÓN" + contenido); } else { JOptionPane.showMessageDialog(null, "NO EXISTEN DATOS POR GUARDAR"); } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 2 + 1$ $M = 3$ $R = 3$

Caso de uso 9: Abrir archivos almacenados en el dispositivo.		
Sentencias	Grafo	Complejidad
<pre> lectura = new FileReader(ubicar); entrada = new BufferedReader(lectura); while ((aux = entrada.readLine()) != null) { cedula = "Cedula: " + aux; nombre = "Nombre: " + entrada.readLine(); sexo = "Sexo: " + entrada.readLine(); fecha = "Fecha Nac: " + entrada.readLine(); telefono = "Teléfono: " + entrada.readLine(); direccion = "Dirección: " + entrada.readLine(); carrera = "Carrera: " + entrada.readLine(); semestre = "Semestre: " + entrada.readLine(); materia = "Materia: " + entrada.readLine(); x = x + cedula + "\n" + nombre + "\n" + sexo + "\n" + fecha + "\n" + telefono + "\n" + direccion + "\n" + carrera + "\n" + semestre + "\n" + materia + "\n"; } entrada.close(); lectura.close(); </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 0 + 1$ $M = 1$ $R = 1$

Caso de uso 10: : Jugar carrera de autos.		
Sentencias	Grafo	Complejidad
<pre> c1 = c.getLblAuto1().getLocation().x; c2 = c.getLblAuto2().getLocation().x; if (c1 < c.getLblMeta().getLocation().x - 10 && c2 < c .getLblMeta().getLocation().x - 10) { etiqueta.setLocation(etiqueta.getLocation().x + 10, etiqueta.getLocation().y); c.repaint(); } else { break; } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 2 + 1$ $M = 3$ $R = 3$

Caso de uso 11: Recuperación de contraseña		
Sentencias	Grafo	Complejidad
<pre> if (CorreoValido) { enviarCorreoElectronico(correoElectronico); mostrarMensajeExito(); } else { mostrarMensajeError(); } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$

Caso de uso 12: Actualización de perfil de usuario		
Sentencias	Grafo	Complejidad
<pre> if (adm.validarEditar(cedula, pgombre, sexo, fechaNac, telefono, direccion, materia, titulo, facultad)) { JOptionPane.showMessageDialog(null, "USUARIO EDITADO"); } else { JOptionPane.showMessageDialog(null, "LLENE TODOS LOS CAMPOS"); } dispose(); } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$

Caso de uso 13: Búsqueda de usuarios		
Sentencias	Grafo	Complejidad
<pre> List<Usuario> buscarUsuarios(String criterioBusqueda) { List<Usuario> usuariosEncontrados = new ArrayList<>(); for (Usuario usuario : usuarios) { if (usuario.coincideCriterioBusqueda(criterioBusqueda)) { usuariosEncontrados.add(usuario); } } return usuariosEncontrados; } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$
Caso de uso 14: Desactivación de cuenta de usuario		
Sentencias	Grafo	Complejidad
<pre> static void desactivarCuentaUsuario() { Scanner scanner = new Scanner(System.in); System.out.print("¿Estás seguro de que deseas desactivar tu cuenta? (S/N): "); String confirmacion = scanner.nextLine(); if (confirmacion.equalsIgnoreCase("S")) { this.isCuentaActiva = false; System.out.println("Tu cuenta ha sido desactivada exitosamente!"); } else { System.out.println("La desactivación de la cuenta ha sido cancelada."); } } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$
Caso de uso 15: Obtener historial de actividad de usuario		
Sentencias	Grafo	Complejidad
<pre> List<ActividadUsuario> obtenerHistorialActividadUsuario(int idUsuario) { List<ActividadUsuario> historialUsuario = new ArrayList<>(); for (ActividadUsuario actividad : historialActividad) { if (actividad.getIdUsuario() == idUsuario) { historialUsuario.add(actividad); } } return historialUsuario; } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$
Caso de uso 16: Gestión de permisos de usuario		
Sentencias	Grafo	Complejidad
<pre> public static void actualizarPermisosUsuario(String usuario, List<String> permisos) { for (String permiso : permisos) { Usuario usuarioEncontrado = buscarUsuario(usuario); if (usuarioEncontrado != null) { for (String permiso : permisos) { usuarioEncontrado.agregarPermiso(permiso); } } else { System.out.println("El usuario " + usuario + " no existe en la base de datos."); } } } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ Desarrollo $M = 2 + 1$ $M = 3$ $R = 3$

Caso de uso 17: Seguimiento de actividad de usuario		
Sentencias	Grafo	Complejidad
<pre> static void mostrarRegistroActividad(String usuario) { System.out.println("Registro de actividad del usuario: " + usuario); Map<String, String> registroActividad = obtenerRegistroActividad(usuario); if (registroActividad.isEmpty()) { System.out.println("No se encontró actividad reciente para el usuario " + usuario); } else { for (Map.Entry<String, String> entry : registroActividad.entrySet()) { String fecha = entry.getKey(); String accion = entry.getValue(); System.out.println("Fecha: " + fecha + " Acción: " + accion); } } } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ <hr/> Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$
Caso de uso 18: Generación de informes de usuarios		
Sentencias	Grafo	Complejidad
<pre> static void generarInformeUsuarios(List<Usuario> usuarios, List<String> camposDatos) { System.out.println("Generando informe de usuarios..."); for (Usuario usuario : usuarios) { System.out.println("Usuario: " + usuario.getNombre()); for (String campo : camposDatos) { String valor = obtenerValorCampo(usuario, campo); System.out.println(campo + ": " + valor); } } } </pre>		$M = NP + 1$ $M = NP + 1$ $M = R$ <hr/> Desarrollo $M = 1 + 1$ $M = 2$ $R = 2$

Los grafos de complejidad generados después de las correcciones y mejoras realizadas en nuestro software han demostrado ser una herramienta invaluable para evaluar el impacto de dichas modificaciones. Mediante la visualización de la complejidad actualizada, hemos podido identificar áreas de mejora y asegurarnos de que nuestras acciones correctivas hayan tenido el efecto deseado. Los grafos nos han proporcionado una visión clara y concisa de la estructura y la complejidad del software, permitiéndonos tomar decisiones informadas para mejorar aún más la calidad y la mantenibilidad del proyecto.

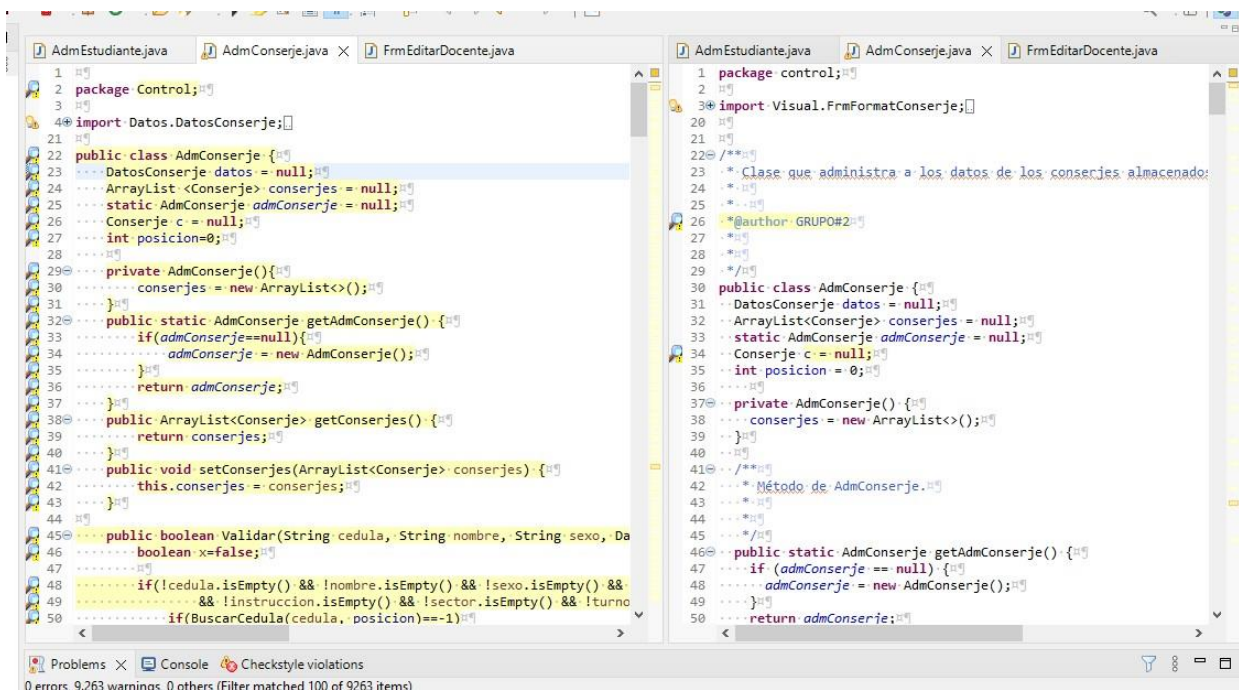
7. Aplicación de Herramientas.

La herramienta Checkstyle para garantizar el cumplimiento de las guías de codificación establecidas por Google. Checkstyle nos ayuda a mantener un estilo de programación consistente y a identificar violaciones de las reglas establecidas. Estos son los pasos que seguimos al aplicar Checkstyle:

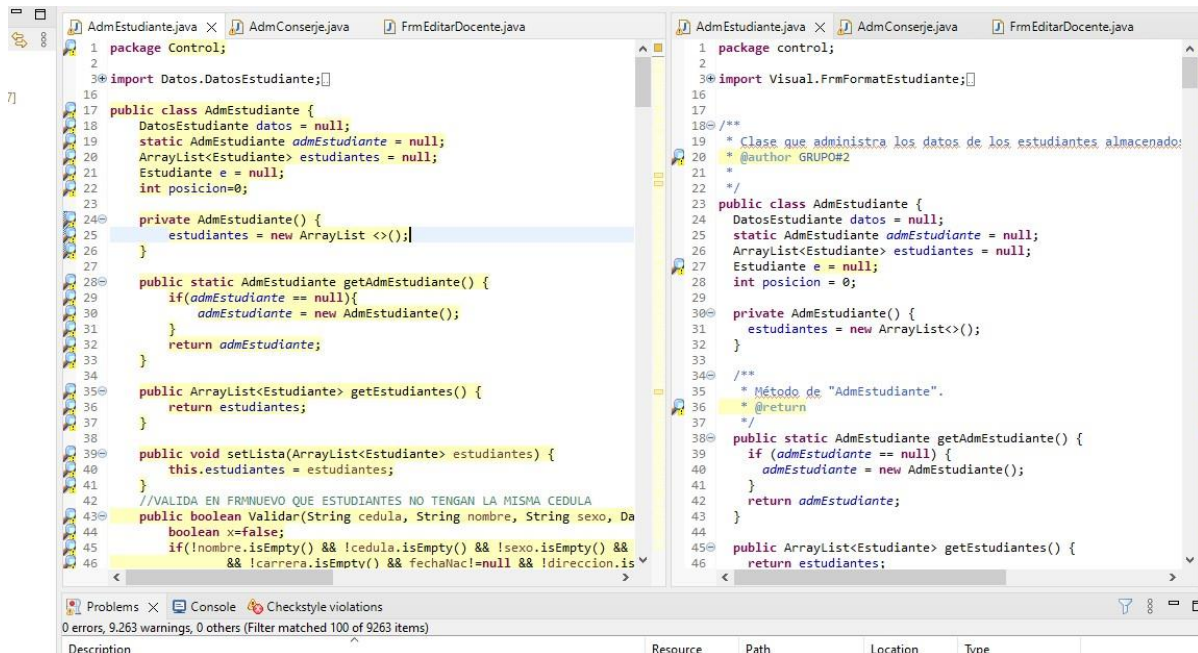
1. Configuramos la herramienta para que aplique las reglas y estándares de codificación definidos por Google, como la indentación, el uso adecuado de espacios en blanco, la longitud de las líneas y el nombramiento de variables y métodos.

2. Ejecutamos Checkstyle en nuestro código fuente para analizarlo y generar informes detallados que muestran las violaciones a las reglas de codificación.
3. Revisamos los informes generados por Checkstyle para identificar las violaciones y las áreas en las que se debe mejorar el cumplimiento de las reglas de codificación.
4. Tomamos las acciones necesarias para corregir las violaciones identificadas por Checkstyle, ajustando el código y aplicando las prácticas de codificación recomendadas.

El uso de Checkstyle nos permite mantener un código limpio, legible y coherente, y nos ayuda a prevenir errores y mejorar la calidad del software. A continuación, se presentan ejemplos visuales que ilustran la aplicación de Checkstyle en el proceso de análisis estático de código.



Código sin aplicación de guías de Java // Código con guías de Java aplicadas



Código sin aplicación de guías de Java // Código con guías de Java aplicadas

Las imágenes destacan la importancia de utilizar herramientas como Checkstyle en el desarrollo de software. Mediante el análisis estático proporcionado por Checkstyle, es posible identificar y corregir violaciones de estilo de codificación, mantener un código limpio y coherente, y mejorar la calidad del software en general. La utilización de Checkstyle como parte de las prácticas de desarrollo contribuye a la estandarización del código, facilita la colaboración entre desarrolladores y ayuda a prevenir errores comunes, lo que resulta en un software más robusto y confiable.

8. Estándares y Guías en el Desarrollo

- **Java Code Conventions.**

Son un conjunto de estándares y convenciones de codificación establecidos por Oracle para el lenguaje de programación Java. Estas convenciones proporcionan directrices sobre cómo escribir código Java legible y consistente, lo que facilita la comprensión, el mantenimiento y la colaboración en proyectos de desarrollo de software.

Algunos aspectos clave de las Java Code Conventions incluyen:

1. **Nomenclatura de variables:** Se recomienda el uso de nombres descriptivos para las variables, utilizando el estilo de escritura camelCase (inicial en minúscula, palabras siguientes en mayúscula).
2. **Formato del código:** Se establece un formato de código estándar para mejorar la legibilidad, incluyendo el uso adecuado de sangrías, espacios en blanco y líneas en blanco.
3. **Convenciones para clases y métodos:** Se sugiere utilizar nombres de clase en formato PascalCase (inicial en mayúscula, palabras siguientes en mayúscula) y nombres de métodos en formato camelCase. Además, se recomienda seguir una estructura de declaración de clases y métodos coherente.
4. **Comentarios y documentación:** Se fomenta el uso de comentarios para explicar el código y su funcionamiento. También se establecen pautas para la documentación Javadoc, que es una forma de generar documentación automática a partir de comentarios especiales en el código fuente.
5. **Gestión de excepciones:** Se definen prácticas recomendadas para la gestión de excepciones, incluyendo la captura y manejo adecuado de excepciones, evitando capturar excepciones demasiado generales.

Es importante seguir estas convenciones en el desarrollo de proyectos Java, ya que facilitan la lectura y comprensión del código, mejoran la colaboración entre desarrolladores y ayudan a mantener un estilo de codificación consistente en el equipo.

- **Java de Google.**

Es un conjunto de guías de estilo y prácticas recomendadas utilizadas por Google en el desarrollo de software en Java. Estas guías buscan promover la legibilidad, mantenibilidad y eficiencia del código, basándose en la vasta experiencia de Google en la creación y mantenimiento de proyectos de gran escala.

Algunos aspectos destacados de Java de Google incluyen:

1. **Nomenclatura de variables y métodos:** Se establece el uso de nombres descriptivos y claros para las variables y métodos, evitando abreviaturas confusas o demasiado cortas. También se recomienda el uso de nombres en formato camelCase.
2. **Formato del código:** Se establecen pautas detalladas para el formato del código, incluyendo el uso de sangrías, espacios en blanco y líneas en blanco. Estas pautas se centran en mejorar la legibilidad y reducir la ambigüedad.
3. **Uso de comentarios:** Se definen prácticas recomendadas para el uso de comentarios, enfatizando su utilidad en la explicación del código, en lugar de simplemente repetir lo que se puede inferir del código en sí.
4. **Estructura de clases y métodos:** Se sugieren pautas para la estructura y organización de las clases y métodos, incluyendo la agrupación lógica de código relacionado y la limitación del tamaño y complejidad de las clases y métodos.
5. **Gestión de excepciones:** Se proporcionan recomendaciones para la gestión de excepciones, incluyendo la especificación de excepciones verificadas en las firmas de los métodos y el manejo adecuado de excepciones no verificadas.

La adopción de las guías de Java de Google puede ayudar a mejorar la calidad y mantenibilidad del código Java, especialmente en proyectos grandes y complejos.

9. Aplicación de V&V en un Programa de Administración de una Escuela.

El programa de administración de la escuela es una aplicación diseñada para facilitar y agilizar la gestión de diferentes roles y procesos dentro de una escuela. Este programa tiene como objetivo principal organizar y centralizar la información relacionada con el personal administrativo, conserjes, secretarías, estudiantes y docentes, brindando una plataforma integrada para realizar diversas tareas administrativas.

En el programa de administración de la escuela se realiza con el propósito de garantizar el cumplimiento de las convenciones de codificación y buenas prácticas, incluyendo las guías de estilo de Java de Google. Checkstyle es una herramienta de análisis estático de código en

Java que permite verificar automáticamente si el código fuente cumple con las reglas y convenciones establecidas.

Al utilizar Checkstyle con las guías de estilo de Java de Google, se asegura que el código del programa de administración de la escuela siga un estándar consistente y coherente. Esto facilita la lectura y el mantenimiento del código, mejora la calidad del software y promueve buenas prácticas de programación en el equipo de desarrollo.

La medición de la complejidad ciclomática es una técnica utilizada para evaluar la complejidad estructural de un programa y determinar el número de caminos de ejecución posibles a través del código. Esta métrica se calcula a partir del grafo de control de flujo del programa y puede ser especialmente útil para identificar partes críticas o complejas del código que pueden requerir una mayor atención y pruebas exhaustivas.

En el contexto del programa de administración de la escuela, se realizaron grafos de casos de uso para representar visualmente el flujo de control en diferentes secciones del código. Estos grafos permiten identificar las diversas rutas de ejecución y ayudan en el cálculo de la complejidad ciclomática.

La complejidad ciclomática se determina contando el número de regiones o nodos en el grafo, donde una región representa una parte del código con un único punto de entrada y salida.

Al calcular la complejidad ciclomática en las partes críticas del código del programa de administración de la escuela, se puede determinar si hay secciones con una complejidad excesiva, lo cual puede indicar un mayor riesgo de errores o dificultades en el mantenimiento y pruebas del software. Identificar estas áreas permite tomar acciones específicas, como simplificar el código, modularizar o refactorizar para reducir la complejidad y mejorar la legibilidad y mantenibilidad.

10. Requerimientos.

10.1. Introducción

10.1.1. Propósito

El propósito del producto es darle al cliente la opción de gestionar a los nuevos usuarios (conserje, empleado y estudiante) que formen parte de su compañía, además, mostrar una lista de usuarios registrados según su rol, guardar la información del usuario dentro del dispositivo (computadora). Se pretende que el proyecto pueda implementar nuevas funcionalidades a futuro.

Dándole así al cliente, la posibilidad de poder llevar un registro de sus empleados e información de sus estudiantes en tiempo real. Adicional, el proyecto mostrará información de la versión del software y un entretenimiento de un juego de carreras.

10.1.2. Alcance

Nuestro equipo está encargado de la realización del sistema de gestión de usuarios junto con la documentación respectiva, con el fin de satisfacer los requerimientos y necesidades del cliente.

Se manejará un tipo de usuario por el momento, el administrador que vendría a ser la secretaria. Soporta una versión de sistema operativo de x86bits y 64bits. Es un programa ligero y sencillo que cuenta con funcionalidades básicas de gestión como: crear, editar, visualizar, eliminar, filtrar. El almacenamiento requiere de un mínimo de 1GB. Muestra una ventana para el inicio de sesión, con el que el usuario tendrá que iniciar sesión para visualizar las funcionalidades del sistema.

10.1.3. Público objetivo y sugerencias de lectura

Este proyecto está destinado a:

- Desarrolladores
- Gerentes del proyecto

- Evaluadores
- Redactores de documentación

La secuencia para leer el documento está definida en la tabla de contenidos.

10.1.4. Definición del producto

El software que se va a desarrollar permitirá la gestión de los usuarios mediante el uso de un computador de escritorio, dándole así al usuario la oportunidad de poder visualizar y gestionar a cualquier hora, los diferentes datos de los usuarios.

Todas estas funcionalidades descritas se podrán realizar de manera correcta, mientras el computador donde el cliente tenga instalado el sistema de gestión de usuarios en cuestión, cuente con almacenamiento dinámico.

10.1.5. Referencias

Las referencias usadas en este documento son las siguientes:

- PROTOCOLO_3_INDUSTRIAS_VF
- Documento Requerimientos Funcionales Libro Operaciones Sept2017_rcg
- Estándar IEEE
- Libro de requerimientos universidad de Loja
- FAOCO-2018-LC025 Documento de alcances y requerimientos funcionales
- Marco normativo
- Red Hat Enterprise Linux 4: Manual de referencia.

10.2. Descripción general

10.2.1. Perspectiva del producto

El producto de software se originó a mejorar y optimizar el manejo de las gestiones de los usuarios, de esta forma permite visualizar información, consultas y registros, además, brinda al usuario la comodidad de poder automatizar y guardar todos registros de forma rápida y

sencilla, directamente desde el sistema. La base de este Sistema es creada referenciando a las funcionalidades que el usuario es capaz de realizar en su computador, además, cuando el usuario inicie sesión en el sistema, se sincronizará la información guardada en bases de datos para poder actualizar algún dato o simplemente tener respaldos seguros de esta información.

10.2.2. Funciones del producto

- El sistema deberá permitir al usuario iniciar sesión.
- El sistema deberá poder gestionar a los diferentes empleados y estudiantes ingresados.
- El sistema deberá mostrar una opción de visualización de todas las personas que están ingresadas.
- El sistema deberá realizar búsquedas de usuarios a través de la cédula.
- El sistema deberá mostrar información Acerca De, de la empresa.
- El sistema permitirá una opción de entretenimiento como es la carrera de autos.

10.2.3. Entorno operativo

La aplicación al desarrollarse con el fin de estar en un computador de escritorio, es compatible tanto para Sistemas operativos Microsoft Windows, las versiones desde la 6.0, dando mucho margen para computadoras antiguas y modernas.

10.2.4. Restricciones de diseño e implementación

RI-1: La aplicación se deberá desarrollar en el lenguaje de Programación Java.

RI-2: Usar un gestor de versiones, en este caso Git.

RI-3: La aplicación deberá pesar menos de 200 Mb, cerca 150 para que no sea pesado para el dispositivo.

RI-4: La velocidad de procedimientos deberá estar optimizada para que no tarde más de 2 segundos en gestionar a los diferentes empleados y estudiantes.

RI-5: El idioma de la aplicación puede ser configurado de inglés a español.

RI-6: Todas las gestiones deberán ser guardadas en la base de datos sin ningún error que perjudique la seguridad de los datos.

10.2.5. Documentación del usuario

Manual de Usuario

1.- Ingreso a la cuenta, el Usuario puede ingresar a su cuenta mediante un usuario y contraseña.

2.- Funciones de pantalla principal: El Usuario podrá realizar las siguientes funciones:

- Agregar, editar y eliminar a un empleado o estudiante
- Visualizar una lista de empleado o estudiante.
- Consultar información por número de cédula.
- Jugar carrera de autos.

10.2.6. Assumptions y dependencias

AS- 1: La aplicación funcionará con internet para conectarse con la base de datos, por lo que el internet o el acceso de datos será necesario.

AS- 2: Para acceder a su cuenta deberá ingresar un usuario y contraseña, y deberá ingresarla cada vez que ingrese a la aplicación por razones de seguridad.

AS- 3: Se asume que la empresa cuenta con un computador donde se instalará el sistema.

DE- 1: La aplicación dependerá de que el usuario inicie sesión para poder acceder a todas las funciones que proporciona esta.

DE- 2: La aplicación dependerá de la base de datos para poder acceder o guardar la información del empleado o estudiante.

DE- 3: La aplicación dependerá de almacenamiento del computador para poder guardar información.

10.3. Requisitos de la interfaz externa

10.3.1. Interfaz de usuario

- Al momento de loggearse a la aplicación deberá ingresar usuario y contraseña, mientras esto no ocurra aparecerá un mensaje de campos vacíos o credenciales incorrectas.

- En cada pantalla de la app aparecerá el nombre de la ventana en la que se encuentra.
- Si los datos que se van a registrar tienen un formato incorrecto o conlleva campos vacíos, el sistema no le permitirá registrarlo hasta llenar todo de forma correcta.
- Se le mostrará una tabla con todos los registros guardados y cantidad de registros.

10.3.2. Interfaz de Hardware

Las cajas de texto permitirán tipiar sus caracteres respectivos, es decir el campo teléfono se debe ingresar solo caracteres numéricos por teclado entonces la caja de texto estará validada de que el usuario tipee sólo caracteres numéricos si presiona otra tecla que no sea número no se le tipiará en la caja de texto. Esta restricción va para todas las cajas de texto.

10.3.3. Interfaz de software

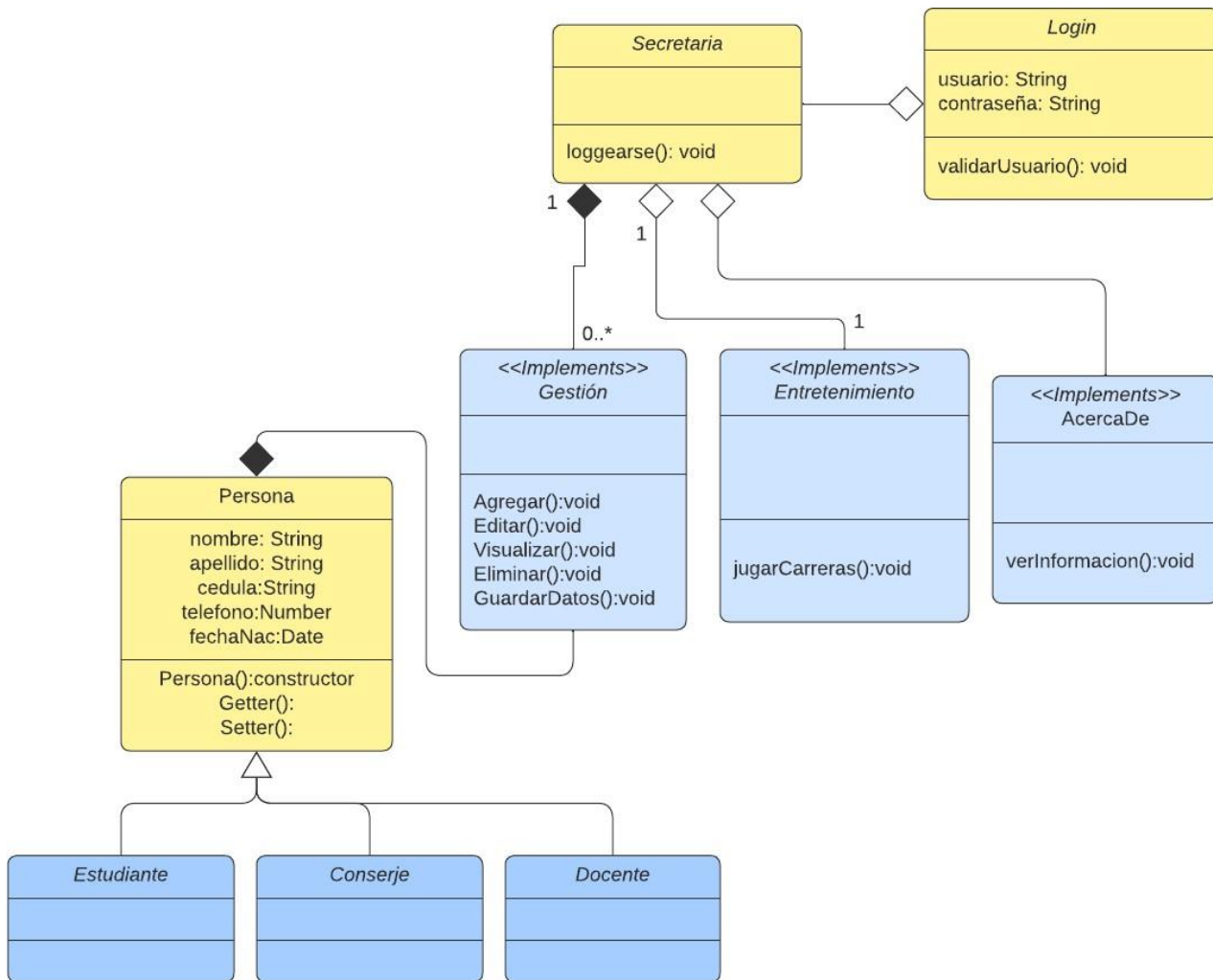
El sistema de gestión de usuarios usará como principal referencia la base de datos.

El software en donde se usará la aplicación será en un computador o laptop.

10.3.4. Interfaces de comunicaciones

La aplicación se comunicará directamente con el servidor a la vez que con la base de datos mediante api rest.

10.4. Modelo de dominio



10.5. System Features (Casos de uso)

10.5.1. Caso de uso 1

Name: Registro de usuario

Goal: Permitir al crear su cuenta en el sistema.

Input: Ingreso de datos.

Output: Mensaje de que su registro fue realizado con éxito.

Main Scenario:**Pre-condition:** Llenar los campos correspondientes**Steps:****Step1:** Ingresar a la página de registro.**Step2:** Registrar datos de usuario en los campos correspondientes (nombre de usuario, contraseña)**Step3:** Dar clic en el botón registrar.**Post-condition:** Dar click en el botón para regresar al menu principal**10.5.2. Caso de uso 2****Name:** Creación de contraseñas seguras**Goal:** Permitir al usuario tener conocimiento de pautas para crear contraseñas seguras**Output:** Mensaje de que la contraseña registrada es segura.**Main Scenario:****Pre-condition:** Llenar los campos correspondientes.**Steps:****Step1:** Ingresar a la página de registro de usuario**Step2:** Llenar sus datos de usuario en los campos correspondientes (nombre de usuario, contraseña)**Step3:** Ingresar contraseña con las pautas establecidas.**Step4:** Dar click en registrar**Post-condition:** Ninguna.**10.5.3. Caso de uso 3****Name:** Ingresar a cuenta**Goal:** Permitir al usuario ingresar a su cuenta.**Input:** Ingreso de datos.**Output:** Mensaje de que su ingreso fue realizado con éxito.

Main Scenario:**Pre-condition:** Llenar los campos correspondientes**Steps:****Step1:** Ingresar a la página de inicio de sesión**Step2:** Llenar sus datos de usuario en los campos correspondientes (nombre de usuario, contraseña)**Step3:** Dar clic en el botón ingresar.**Post-condition:** Dar click en el botón para regresar al menu principal**10.5.4. Caso de uso 4****Name:** Gestionar cuentas**Goal:** Permitir al usuario gestionar los datos de empleados y estudiantes en el sistema.**Input:** Agregar, editar, visualizar y eliminar.**Output:** Mensaje de confirmación de haber agregado, editado, mostrado y eliminado a un empleado o estudiante.**Main Scenario:****Pre-condition:** Haber iniciado sesión el usuario.**Steps:****Step1:** Ingresar a su cuenta.**Step2:** Escoger una opción del menú (Nuevo, editar, visualizar o eliminar).**Step3:** Seleccionar estudiante, docente o conserje a gestionar.**Step4:** Si se edita a un empleado o estudiante se requiere buscar por número de cédula antes.**Step5:** Si se elimina a un empleado o estudiante se requiere seleccionarlo de una lista antes.**Post-condition:** Ninguna.**10.5.5. Caso de uso 5****Name:** Filtrar por edad y género

Goal: Permitir al usuario buscar por la edad y género a uno o varios empleados o estudiantes agregados en la base de datos.

Input: Ingresar la edad y el género.

Output: Mensaje de que el filtro por edad y género fue exitoso o que no existen datos al respecto.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Ingresar a su cuenta.

Step2: Dar clic en la opción filtrar.

Step3: Ingresar una edad y seleccionar un género.

Step4: Dar clic en filtrar.

Post-condition: Ninguna.

10.5.6. Caso de uso 6

Name: Guardar los datos en el dispositivo.

Goal: Permitir al usuario guardarlos datos de un empleado o estudiante en su computadora o laptop.

Input: Seleccionar la opción de guardar datos.

Output: Mostrar una interfaz donde se guardarán los datos en formato de lectura.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Ingresar a su cuenta.

Step2: Agregar un nuevo empleado o estudiante.

Step3: Dar clic en guardar.

Step4: Seleccionar la opción de guardar datos en el dispositivo ubicado en el menú.

Step5: Se abrirá una interfaz, agregar un nombre al documento a guardar en el dispositivo. Se guardará la información con extensión. mabel

Post-condition: Ninguna.

10.5.7. Caso de uso 7

Name: Gestión de roles de permiso.

Goal: Permitir al administrador asignar roles y permisos a los usuarios.

Input: Seleccionar la opción de asignar roles.

Output: Mostrar una interfaz donde se podrá modificar los roles y permisos.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Ingresar a su cuenta.

Step2: Seleccionar la opción asignar roles y permisos.

Step3: Asignar roles y permisos a los usuarios.

Step4: Dar click en guardar.

Post-condition: Ninguna

10.5.8. Caso de uso 8

Name: Gestion de archivos.

Goal: Permitir al usuario almacenar archivos.

Input: Seleccionar la opción de guardar datos.

Output: Mostrar una interfaz donde se cargará el archivo.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Ingresar a su cuenta.

Step2: Seleccionar la opción abrir que se encuentra dentro del menú.

Step3: Seleccionar la ruta donde se cargará el archivo en formato. mabel

Post-condition: Ninguna

10.5.9. Caso de uso 9

Name: Abrir archivos almacenados en el dispositivo.

Goal: Permitir al usuario abrir archivos almacenados de un empleado o estudiante.

Input: Seleccionar la opción de abrir datos.

Output: Mostrar una interfaz donde se seleccionará el archivo a abrir.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Ingresar a su cuenta.

Step2: Seleccionar la opción abrir que se encuentra dentro del menú.

Step3: Seleccionar la ruta donde se encuentra el archivo a abrir con formato. mabel

Post-condition: Ninguna

10.5.10. Caso de uso 10

Name: Jugar carrera de autos.

Goal: Permitir al usuario entretenerse con un juego de carreras.

Input: Dar inicio a la partida.

Output: Mensaje donde aparece el número del carro ganador.

Main Scenario:

Pre-condition: Haber iniciado sesión.

Steps:

Step1: Seleccionar en el menú la opción de Juego

Step2: Dar clic en el botón iniciar carrera.

Post-condition: Ninguna

10.5.11. Caso de uso 11

Name: Recuperación de contraseña

Goal: Permitir a los usuarios recuperar su contraseña en caso de olvido.

Input: Correo electrónico registrado.

Output: Correo electrónico con instrucciones para restablecer la contraseña.

Main Scenario:

Pre-condition: El usuario debe tener una cuenta registrada.

Steps:

Step1: El usuario accede a la página de recuperación de contraseña.

Step2: El usuario ingresa su correo electrónico registrado.

Step3: El sistema verifica si el correo electrónico está asociado a una cuenta válida.

Step4: Si el correo electrónico es válido, el sistema envía un correo electrónico al usuario con instrucciones para restablecer la contraseña.

Step5: El usuario sigue las instrucciones del correo electrónico para restablecer la contraseña.

Post-condition: El usuario recibe un correo electrónico con instrucciones para restablecer la contraseña.

10.5.12. Caso de uso 12

Name: Actualización de perfil de usuario

Goal: Permitir a los usuarios actualizar su información de perfil.

Input: Datos actualizados del usuario.

Output: Confirmación de actualización de perfil exitosa.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de perfil.

Step2: El usuario modifica los campos deseados en su perfil (nombre, dirección, información de contacto, etc.).

Step3: El usuario hace clic en el botón de guardar cambios.

Step4: El sistema valida los datos ingresados.

Step5: El sistema actualiza la información de perfil del usuario.

Step6: El sistema muestra un mensaje de confirmación

Post-condition: Ninguna

10.5.13. Caso de uso 13

Name: Búsqueda de usuarios

Goal: Permitir a los usuarios buscar otros usuarios en el sistema.

Input: Criterios de búsqueda (nombre de usuario, ubicación, intereses, etc.).

Output: Lista de usuarios que coinciden con los criterios de búsqueda.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de búsqueda de usuarios.

Step2: El usuario ingresa los criterios de búsqueda deseados.

Step3: El usuario hace clic en el botón de buscar.

Step4: El sistema busca en la base de datos de usuarios y encuentra aquellos que coinciden con los criterios de búsqueda.

Step5: El sistema muestra una lista de usuarios encontrados.

Post-condition: El usuario ve la lista de usuarios que coinciden con los criterios de búsqueda

10.5.14. Caso de uso 14

Name: Desactivación de cuenta de usuario

Goal: Permitir a los usuarios desactivar su cuenta en el sistema.

Input: Confirmación de desactivación de cuenta.

Output: Cuenta de usuario desactivada.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de configuración de cuenta.

Step2: El usuario selecciona la opción de desactivar cuenta.

Step3: El usuario confirma su intención de desactivar la cuenta.

Step4: El sistema desactiva la cuenta del usuario.

Step5: El sistema muestra un mensaje de confirmación de la desactivación de la cuenta.

Post-condition: La cuenta del usuario se desactiva y ya no puede acceder al sistema.

10.5.15. Caso de uso 15

Name: Historial de actividad de usuario

Goal: Permitir a los usuarios ver su historial de actividad en el sistema.

Input: N/A

Output: Historial de actividad del usuario.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de historial de actividad.

Step2: El sistema muestra una lista ordenada cronológicamente de las acciones realizadas por el usuario (inicio de sesión, cambios en el perfil, etc.).

Post-condition: El usuario visualiza su historial de actividad en el sistema.

10.5.16. Caso de uso 16

Name: Gestión de permisos de usuario

Goal: Permitir a los administradores asignar y modificar los permisos de un usuario.

Input: Selección del usuario y asignación/modificación de permisos.

Output: Confirmación de cambios en los permisos del usuario.

Main Scenario:

Pre-condition: El usuario debe tener privilegios de administrador.

Steps:

Step1: El administrador accede a la página de gestión de permisos de usuario.

Step2: El administrador selecciona un usuario al que desea asignar o modificar permisos.

Step3: El administrador elige los permisos que desea asignar o modificar para el usuario.

Step4: El administrador confirma los cambios.

Step5: El sistema actualiza los permisos del usuario seleccionado.

Step6: El sistema muestra un mensaje de confirmación de los cambios realizados

Post-condition: Los permisos del usuario se actualizan según las selecciones del administrador.

10.5.17. Caso de uso 17

Name: Seguimiento de actividad de usuario

Goal: Permitir a los administradores y usuarios rastrear y revisar la actividad reciente de un usuario.

Input: Selección del usuario y visualización de la actividad.

Output: Registro de actividad del usuario.

Main Scenario:

Pre-condition: El usuario o administrador debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario o administrador accede a la página de seguimiento de actividad de usuario.

Step2: El usuario o administrador selecciona el usuario específico del que desea ver la actividad.

Step3: El sistema muestra un registro de las acciones recientes realizadas por ese usuario (inicio de sesión, modificaciones de perfil, etc.).

Post-condition: El usuario o administrador visualiza la actividad reciente del usuario seleccionado.

10.5.18. Caso de uso 18

Name: Generación de informes de usuarios

Goal: Permitir a los administradores generar informes y estadísticas relacionadas con los usuarios del sistema.

Input: Configuración del informe (rango de fechas, criterios de selección, etc.).

Output: Informe generado con los datos solicitados.

Main Scenario:

Pre-condition: El usuario debe tener privilegios de administrador.

Steps:

Step1: El administrador accede a la página de generación de informes de usuarios.

Step2: El administrador configura los parámetros del informe, como el rango de fechas, los criterios de selección, los campos de datos requeridos, etc.

Step3: El administrador solicita la generación del informe.

Step4: El sistema recopila los datos relevantes de los usuarios según los parámetros configurados.

Step5: El sistema genera el informe con los datos recopilados.

Step6: El sistema muestra el informe generado, que puede ser descargado o visualizado en línea.

Post-condition: El administrador obtiene un informe con los datos de los usuarios según la configuración realizada.

10.5.19. Caso de uso 19

Name: Exportación de datos de usuario

Goal: Permitir a los usuarios exportar sus datos personales del sistema.

Input: Solicitud de exportación de datos.

Output: Archivo descargable con los datos personales del usuario.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de configuración de exportación de datos.

Step2: El usuario selecciona los tipos de datos que desea exportar (perfil, historial de actividad, mensajes, etc.).

Step3: El usuario solicita la exportación de los datos.

Step4: El sistema recopila los datos seleccionados del usuario.

Step5: El sistema genera un archivo descargable (por ejemplo, CSV) con los datos exportados.

Step6: El sistema muestra un enlace de descarga al archivo exportado.

Post-condition: El usuario puede descargar el archivo con sus datos personales exportados.

10.5.20. Caso de uso 20

Name: Gestión de preferencias de idioma

Goal: Permitir a los usuarios seleccionar su preferencia de idioma en el sistema.

Input: Selección del idioma preferido.

Output: Confirmación del cambio de idioma en el sistema.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de gestión de preferencias de idioma.

Step2: El usuario selecciona su idioma preferido de una lista de opciones disponibles.

Step3: El usuario guarda los cambios realizados.

Step4: El sistema actualiza el idioma del usuario en el sistema.

Step5: El sistema muestra un mensaje de confirmación del cambio de idioma.

Post-condition: El idioma preferido del usuario se actualiza en el sistema y se muestra en la interfaz de usuario.

10.5.21. Caso de uso 21

Name: Gestión de preferencias de privacidad

Goal: Permitir a los usuarios establecer y modificar sus preferencias de privacidad en el sistema.

Input: Configuración de las preferencias de privacidad.

Output: Confirmación de los cambios en las preferencias de privacidad del usuario.

Main Scenario:

Pre-condition: El usuario debe haber iniciado sesión en el sistema.

Steps:

Step1: El usuario accede a la página de gestión de preferencias de privacidad.

Step2: El usuario configura sus preferencias de privacidad, como quién puede ver su perfil, recibir notificaciones, compartir información, etc.

Step3: El usuario guarda los cambios realizados.

Step4: El sistema actualiza las preferencias de privacidad del usuario.

Step5: El sistema muestra un mensaje de confirmación de los cambios realizados en las preferencias de privacidad.

Post-condition: Las preferencias de privacidad del usuario se actualizan según las selecciones realizadas.

10.6. Requisitos Funcionales

RF-01	Se podrá iniciar sesión mediante usuario y contraseña.
RF-02	Una vez el usuario se haya loggeado, podrá gestionar los datos de empleados y estudiantes.
RF-03	El inicio de sesión sólo valida usuario y contraseña creados dentro del código, no permite que cualquiera pueda ingresar ni crear un usuario.
RF-04	El sistema permite ingresar un nuevo usuario, editarlo, visualizarlo o eliminarlo.
RF-05	El sistema permite editar los datos de los empleados y estudiantes por medio de su número de cédula.
RF-06	El sistema filtra datos por edad y género.
RF-07	El sistema permite abrir archivos ubicados en el dispositivo, pero con formato. mabel.
RF-08	El sistema permite guardar archivos dentro del dispositivo, pero con formato. mabel, para sólo lectura.
RF-09	El sistema permitirá el entretenimiento por lo que se podrá jugar carrera de autos.
RF-10	El sistema mostrará información sobre su creador y la versión.

10.7. Requerimientos No Funcionales

RNF-01	La aplicación se desarrollará con el lenguaje de programación Java.
--------	---

RNF-02	El servicio debe estar disponible en las 8 horas laborables de la compañía .
RNF-03	El sistema debe soportar más de 1 gestión por minuto.
RNF-04	Los tiempos de respuesta deben ser menor a 3000ms.

10.8. Otros requisitos no funcionales

10.8.1. Requisitos de desempeño

- Al momento de iniciar la carga de la aplicación, este no debe demorar más de 5 seg.
- El 95% de las funciones del sistema de gestión de usuarios, deberán ejecutarse en menos de 1 minuto.
- La instalación del sistema en el computador o laptop, no ocupará un alto rendimiento en el mismo.

10.8.2. Safety Requirements

- La aplicación solo podrá ser ejecutada desde una computadora de escritorio o laptop.
- Debe ser instalado por la empresa de software, debido a que si se la comparte por cualquier persona a través de cualquier podría haber suplantación de información.
- La aplicación puede presentar problemas si el dispositivo en donde está instalada tiene problemas de almacenamiento interno.

10.8.3. Security Requirements

- El sistema debe desarrollarse aplicando patrones y recomendaciones de programación que incrementen la seguridad de datos.
- El sistema debe cerrar sesión luego de máximo 30 minutos de inactividad.
- La aplicación debe cumplir con controles de seguridad relacionados a su utilización a través de redes públicas y privadas, garantizando la confidencialidad, integridad y disponibilidad de la información y acceso a ella.
- Implementar controles para evitar el acceso de usuarios no reconocidos en el sistema.
- Validar los datos que se reciben y velar por la integridad de los datos que se devuelven.
- Se debe desarrollar el control de sesiones seguras y validación de datos, desinfección y eliminación de códigos de depuración (Debugging Codes).

10.8.4. Atributos de calidad del software

- Analizar riesgos y vulnerabilidades del entorno de despliegue del cliente atendiendo a sus características.
- Almacenar los mensajes intercambiados en ficheros logs para su posterior consulta.

10.9. Otros requerimientos

Apéndice A: Glosario

Login: La interfaz donde el usuario que gestionar a los usuarios, deberá iniciar sesión a través de un usuario y contraseña.

Sistema operativo: Es un software que coordina y dirige todos los servicios y aplicaciones que usa el usuario en una computadora.

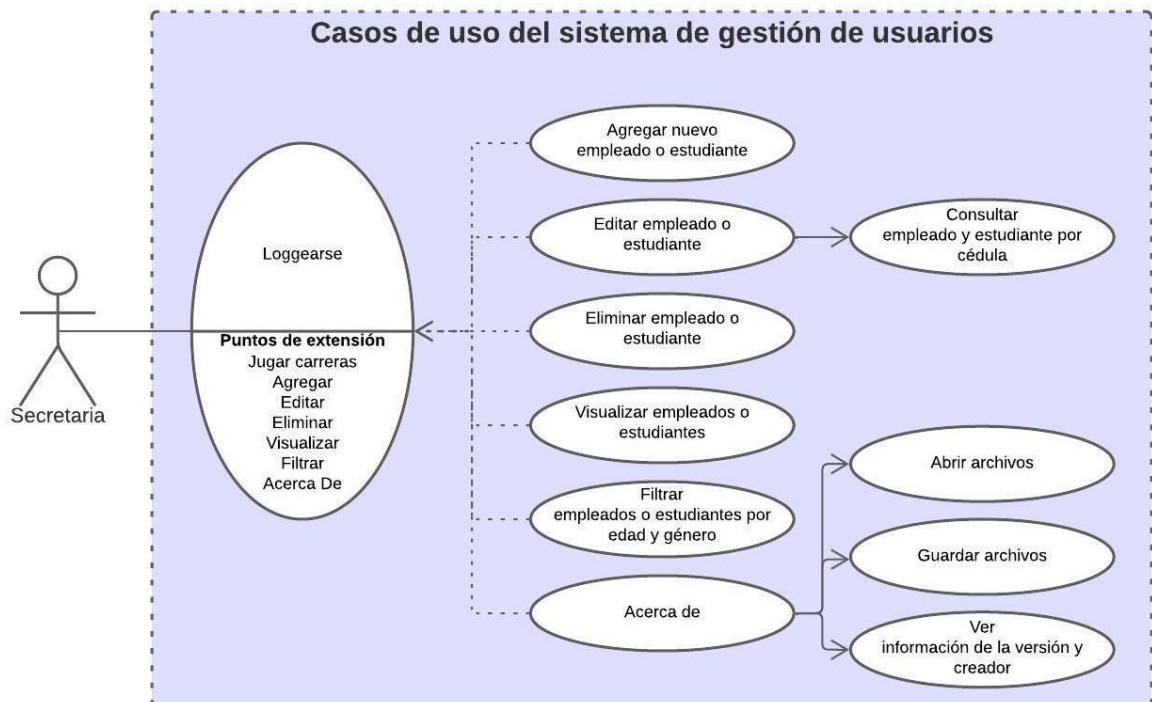
Interfaz: Conexión física y funcional entre dos aparatos, dispositivos o sistemas que funcionan independientemente uno del otro.

Usuario: Persona que utiliza un producto software. Nombre con el que se inicia sesión.

Loggeado: Acción de haber iniciado sesión.

Apéndice B: Modelo de Análisis

Diagrama de casos de uso



Apéndice C: To Be Determined List

http://www.fao.org/fileadmin/user_upload/FAO-countries/Colombia/docs/Junio_2018/LC025/FAOCO-2018-LC025_Documento_de_Alcances_y_Requerimientos_Funcionales.pdf

<https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rq-es-4/s1-intro-conventions.html>

http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:pis:ejemplo_de_especificacion_de_requerimientos_-_para_sesion_9.pdf

http://dspace.utpl.edu.ec/bitstream/123456789/1510/3/Utpl_Valdivieso_Narv%C3%A1ez_Daniel_Estivan_005x1109.pdf

<https://www.sciencedirect.com/topics/engineering/safety-requirement#:~:text=The%20safety%20requirements%20are%20those,redution%20of%20a%20given%20risk.>

http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992018000500015&lng=es&nrm=iso

11. Manual Técnico.

11.1. Objetivos

Desarrollar un software de gestión de usuarios que permita al cliente administrar y registrar a nuevos usuarios (conserje, empleado y estudiante) en tiempo real, así como acceder a una lista de usuarios registrados según su rol, almacenando la información de manera segura en el dispositivo (computadora). Además, el software deberá tener capacidad para implementar futuras funcionalidades.

11.2. Objetivos Específicos.

- Aumentar la eficiencia en el manejo de los recursos humanos y materiales
- Registrar y gestionar nuevos usuarios (conserje, empleado y estudiante).

- Mostrar una lista de usuarios registrados según su rol.
- Almacenar de forma segura la información de los usuarios en el computador del cliente.

11.3. Alcance

Nuestro equipo está encargado de la realización del sistema de gestión de usuarios junto con la documentación respectiva, con el fin de satisfacer los requerimientos y necesidades del cliente.

Se manejará un tipo de usuario por el momento, el administrador que vendría a ser la secretaria. Soporta una versión de sistema operativo de x86bits y 64bits. Es un programa ligero y sencillo que cuenta con funcionalidades básicas de gestión como: crear, editar, visualizar, eliminar, filtrar. El almacenamiento requiere de un mínimo de 1GB. Muestra una ventana para el inicio de sesión, con el que el usuario tendrá que iniciar sesión para visualizar las funcionalidades del sistema.

11.4. Requerimientos Mínimos de Hardware.

Procesador: CORE

Memoria RAM (Mínimo): 1GB

Disco Duro: 20GB

11.5. Requerimientos Mínimos de Software.

Privilegios de Administrador: Si/No

Sistema Operativo: Windows 7/8/10/11

11.6. Herramientas Utilizadas para el Desarrollo.

Netbeans

NetBeans es una herramienta versátil y potente que brinda a los programadores un entorno de desarrollo completo y amigable para crear aplicaciones en diversos lenguajes y tecnologías. Su amplia gama de características y su comunidad activa hacen de NetBeans una elección popular entre los desarrolladores de software.

SQL Server ofrece escalabilidad, seguridad robusta, alta disponibilidad y funcionalidad completa, lo que lo convierte en una opción confiable y poderosa para proyectos de desarrollo de software.

Eclipse Java

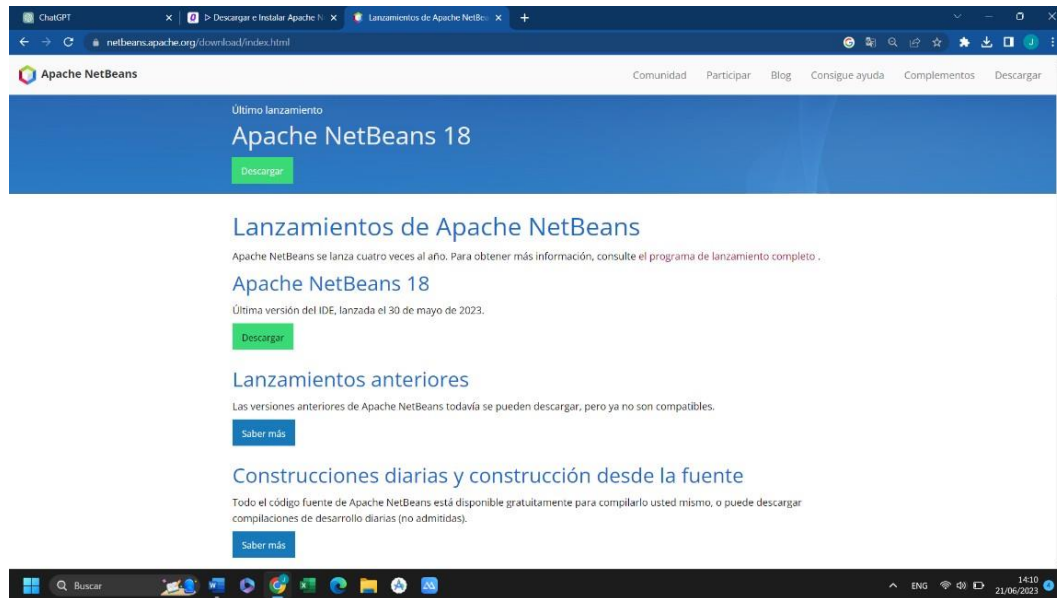
Eclipse Java es un IDE muy popular y ampliamente utilizado para el desarrollo de aplicaciones en Java. Proporciona herramientas y funcionalidades robustas que ayudan a los programadores a escribir y depurar código Java de manera eficiente, y su amplia comunidad y ecosistema de complementos lo convierten en una opción sólida para los desarrolladores de Java.

Jdk

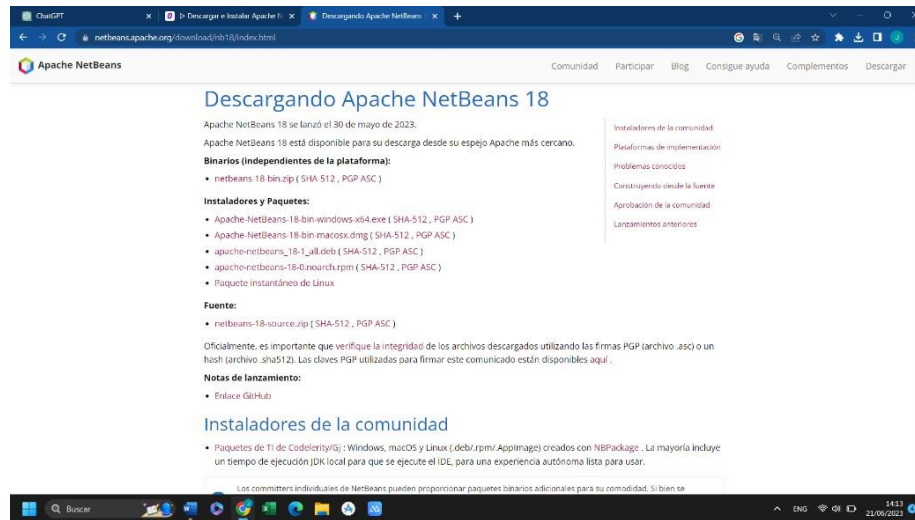
JDK es un conjunto de herramientas esenciales para desarrollar, compilar y ejecutar aplicaciones Java. Proporciona el compilador de Java, bibliotecas de clases, el JRE y varias herramientas de desarrollo que ayudan a los programadores a crear aplicaciones Java de manera eficiente.

11.7. Instalación

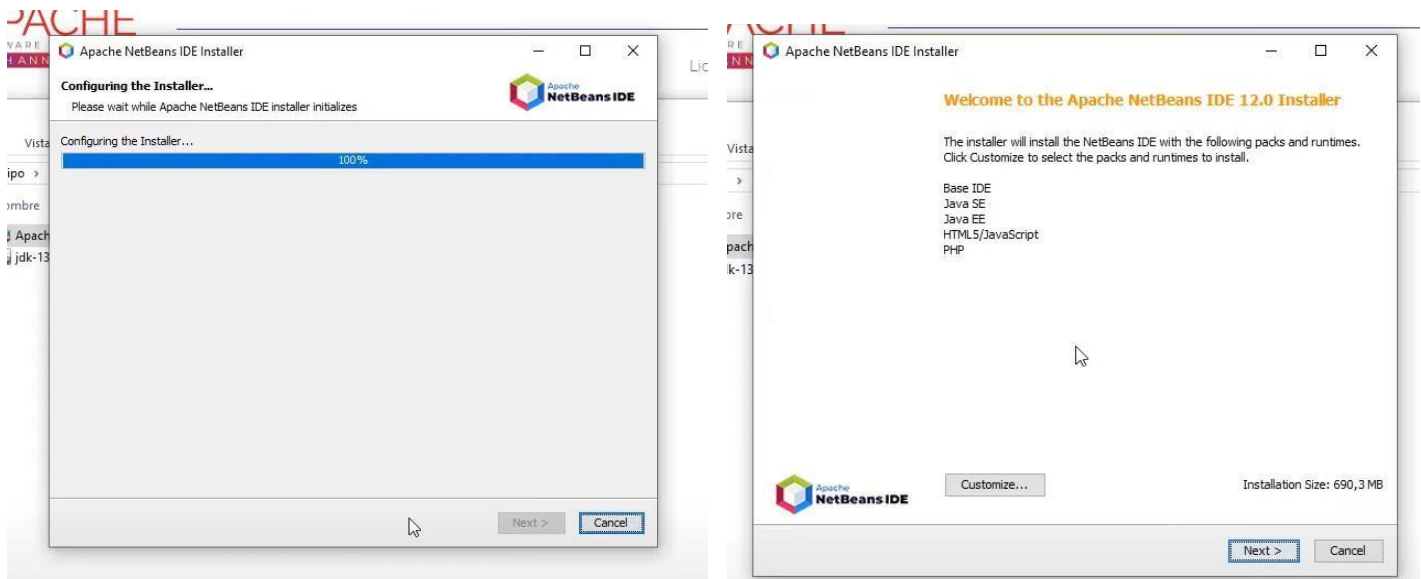
1. Descargar de la página oficial de Apache Netbeans



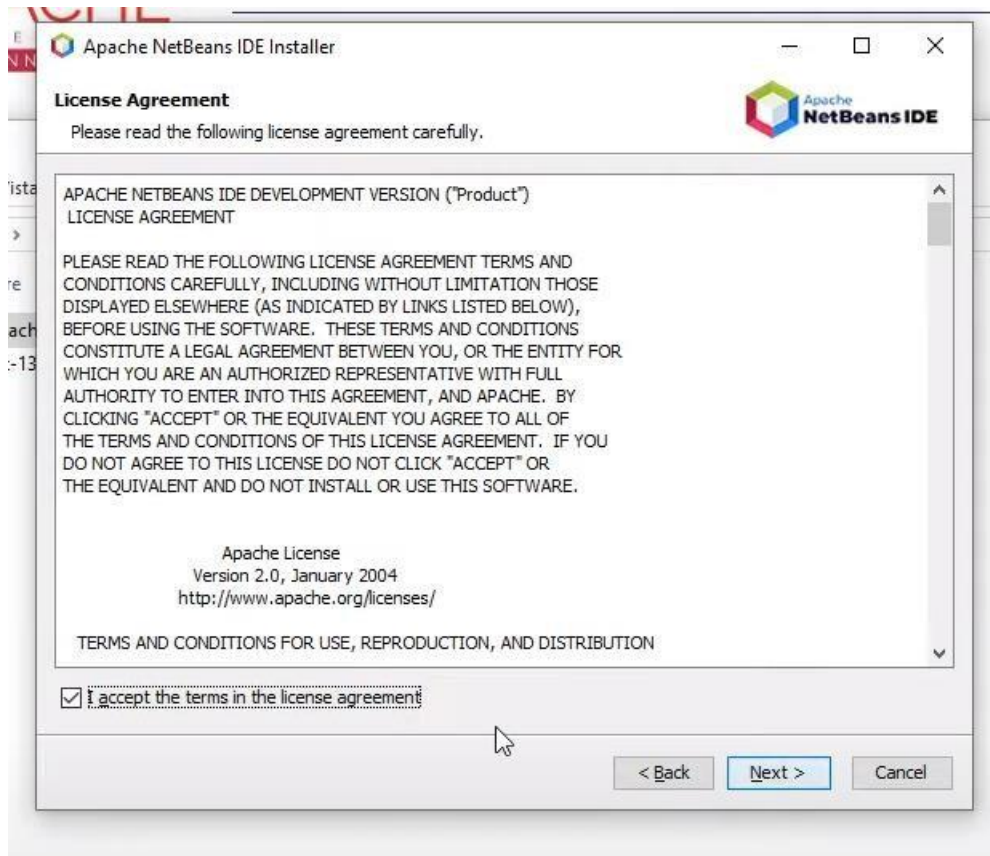
2. Seleccionar el instalador del programa NetBeans 18 y damos clic



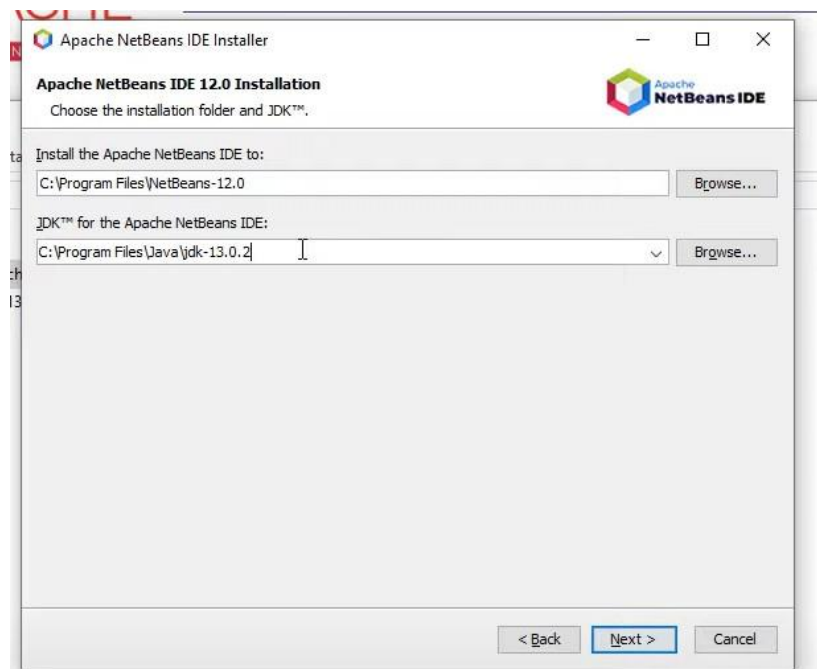
3. Ejecutamos Netbeans



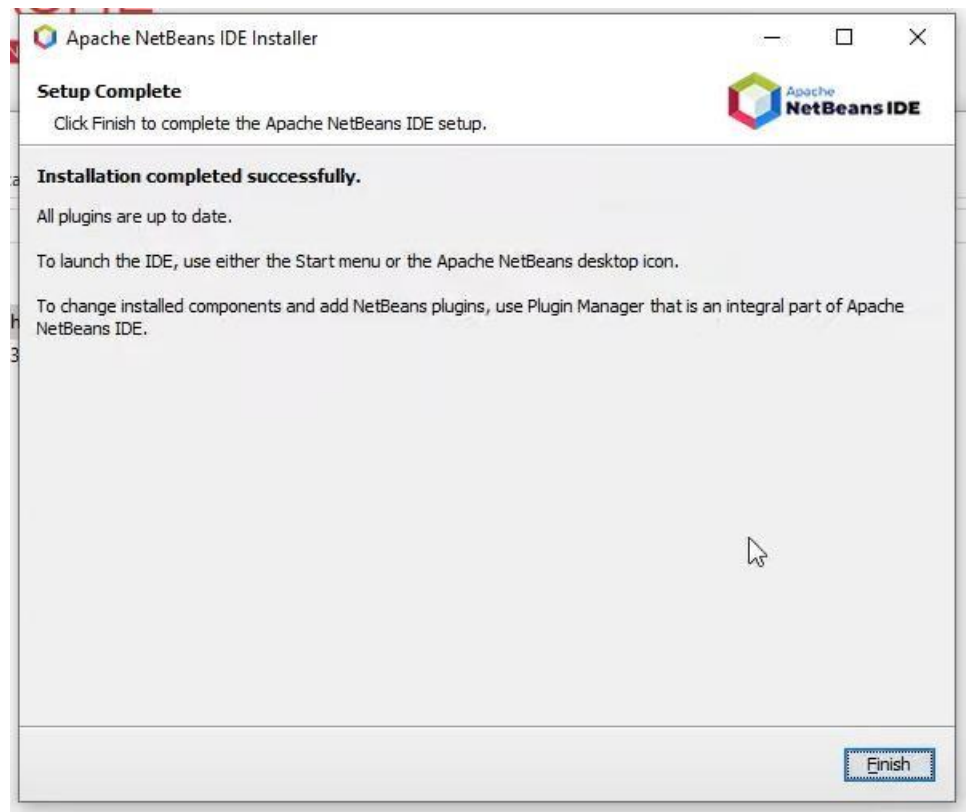
4. Aceptamos la licencia y damos en siguiente para seguir con el proceso de instalacion



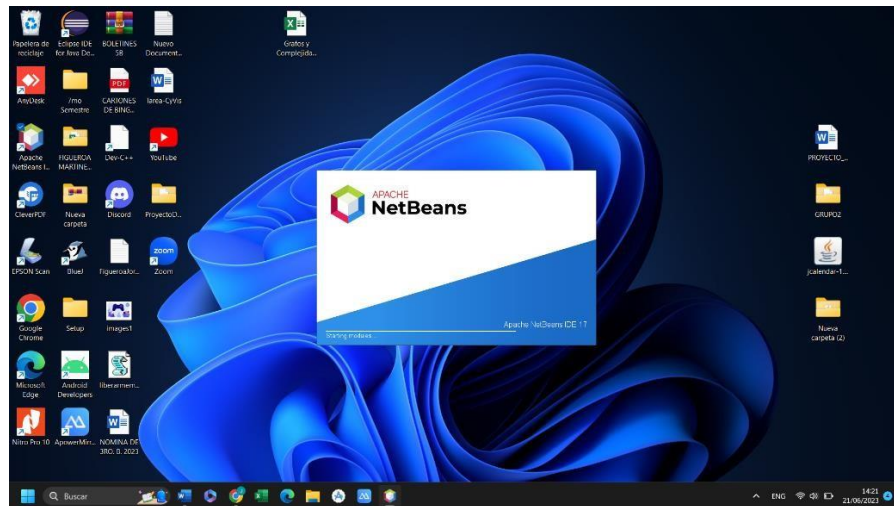
5. El instalador busca donde se encuentra ubicado el JDK

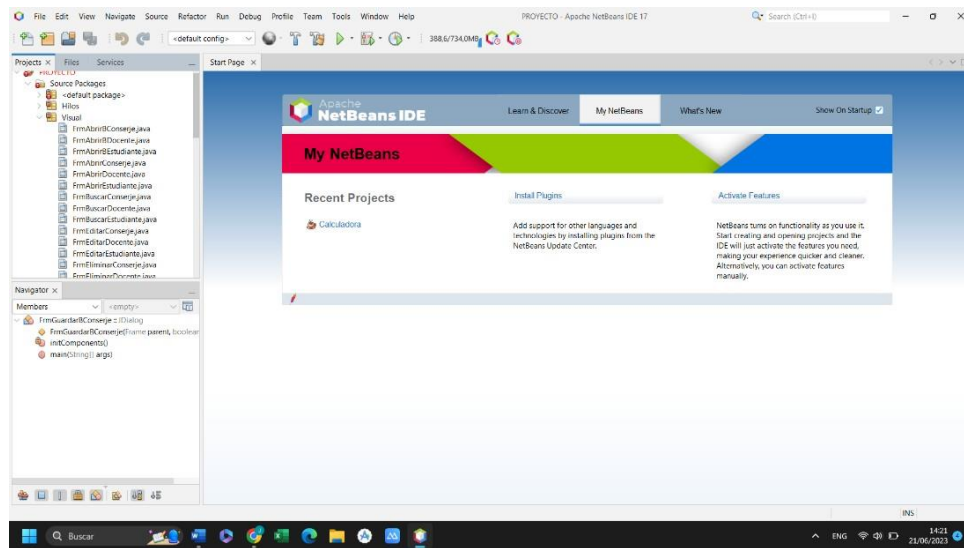


6. Termina la instalación



7. Abrimos NetBeans para proceder con la ejecución de cualquier proyecto





11.8. Guía de Uso

1. ***Ejecutamos el proyecto y se nos abrirá un Login en el cual podremos a escribir el User y el Password.***

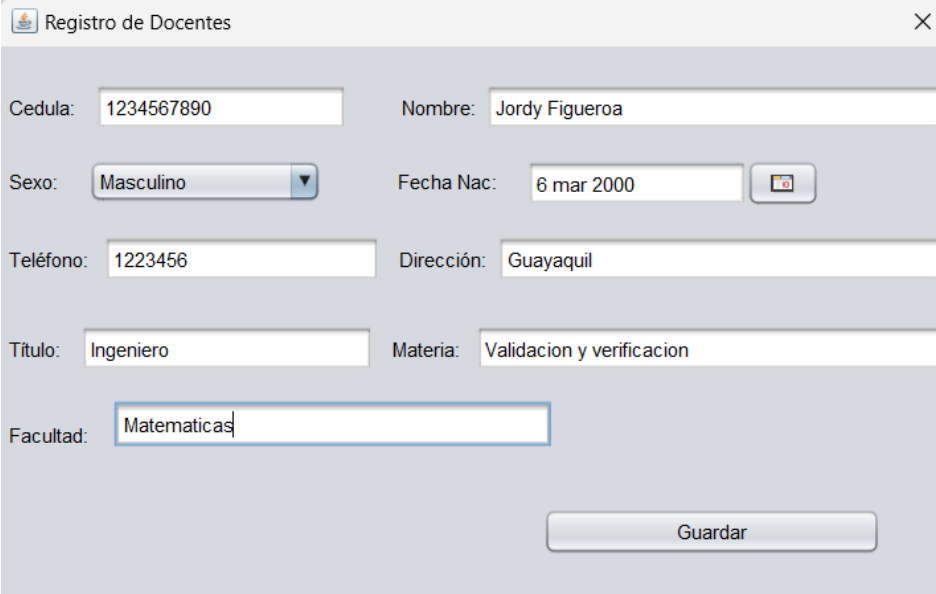
2. ***Ingresamos al menú principal del aplicativo donde tendremos distintas opciones como crear, búsqueda y eliminar.***



3. La persona con el rol de secretaria podrá Agregar, editar y visualizar un docente, estudiante y conserje.



4. Para agregar un nuevo docente se da clic y se abre la siguiente ventana.



Registro de Docentes

Cedula: 1234567890 Nombre: Jordy Figueroa

Sexo: Masculino Fecha Nac: 6 mar 2000

Teléfono: 1223456 Dirección: Guayaquil

Título: Ingeniero Materia: Validacion y verificacion

Facultad: Matematicas

Guardar

5. Para editar un docente se da clic en Editar y se abre una ventana de búsqueda por Cedula.



Edicion de un Docente

Cedula: 1234567890 Nombre: Jordy Figueroa

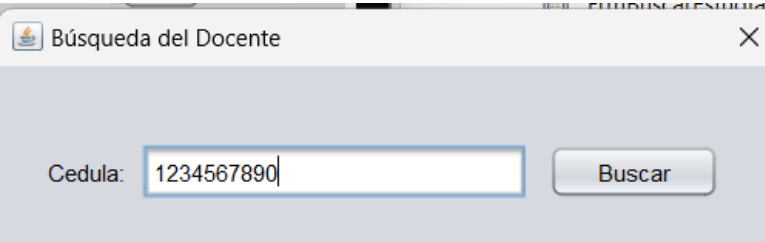
Sexo: Masculino Fecha Nac: 6 mar 2000

Teléfono: 1223456 Dirección: Guayaquil

Título: Ingeniero Materia: Validacion y verificacion

Facultad: Matematicas

Editar

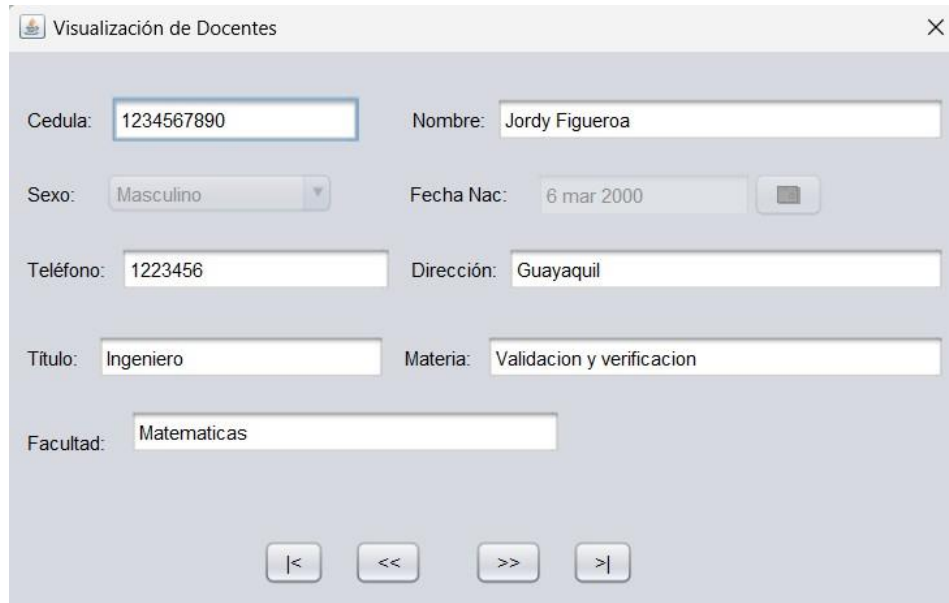


Búsqueda del Docente

Cedula: 1234567890

Buscar

6. **En la ventana de visualización de docente procedemos a revisar los docentes registrados.**



The screenshot shows a window titled "Visualización de Docentes" with a close button (X) in the top right corner. The window contains a form with the following fields:


- Cedula: 1234567890
- Nombre: Jordy Figueroa
- Sexo: Masculino (dropdown menu)
- Fecha Nac: 6 mar 2000
- Teléfono: 1223456
- Dirección: Guayaquil
- Título: Ingeniero
- Materia: Validacion y verificacion
- Facultad: Matematicas

At the bottom of the window, there are four navigation buttons: |<, <<, >>, and >|.

7. **Para registrar un estudiante vamos a la sección de secretaria y damos clic en nuevo -> Estudiante**



8. ***Se abrirá una ventana para registrar los datos del estudiante y elegir la carrera y el semestre.***

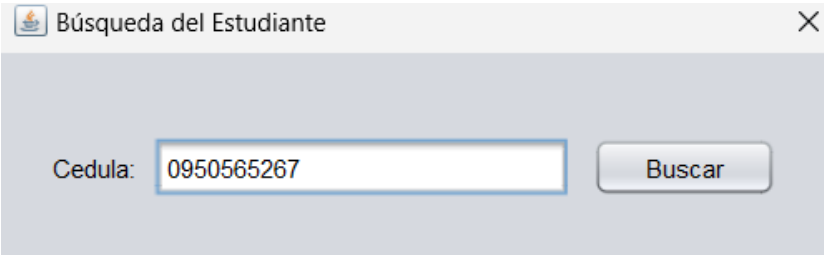


A screenshot of a software window titled "Registro de Estudiantes". It contains several input fields for student data: Cedula (0950565267), Nombre (Jordy Figueroa), Sexo (Masculino), Fecha Nac (6 mar 2000), Teléfono (123456), Dirección (Duran), Carrera (Software), Semestre (Octavo), and Materia (acion y validacion de software). A "Guardar" button is at the bottom right.

Cedula:	0950565267	Nombre:	Jordy Figueroa
Sexo:	Masculino	Fecha Nac:	6 mar 2000
Teléfono:	123456	Dirección:	Duran
Carrera:	Software	Semestre:	Octavo
Materia:	acion y validacion de software		

Guardar

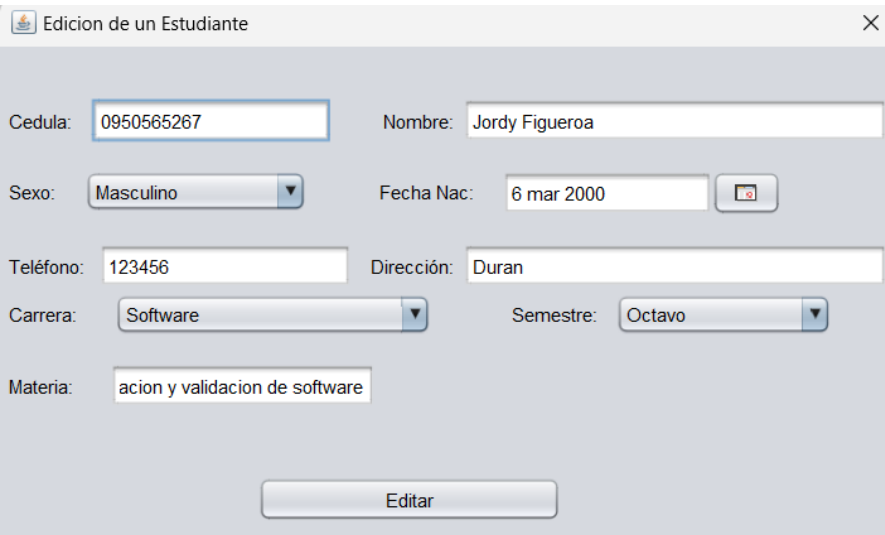
9. ***Para editar el estudiante procedemos a dar clic en secreteria -> Editar -> Estudiante y se abrirá una ventana para ingresar la cedula y editar los datos***



A screenshot of a software window titled "Búsqueda del Estudiante". It has a single input field for "Cedula" containing "0950565267" and a "Buscar" button.

Cedula:	0950565267
---------	------------

Buscar



A screenshot of a software window titled "Edicion de un Estudiante". It contains the same input fields as the registration window, with the "Cedula" field highlighted. An "Editar" button is at the bottom center.

Cedula:	0950565267	Nombre:	Jordy Figueroa
Sexo:	Masculino	Fecha Nac:	6 mar 2000
Teléfono:	123456	Dirección:	Duran
Carrera:	Software	Semestre:	Octavo
Materia:	acion y validacion de software		

Editar

10. Para visualizar un estudiante vamos a Secretaría -> visualizar y seleccionamos estudiante

The image shows a window titled 'Visualización de Estudiantes'. It contains a form with the following fields and values: 'Cedula:' 0950565267, 'Nombre:' Jordy Figueroa, 'Sexo:' Masculino (dropdown), 'Fecha Nac:' 6 mar 2000 (calendar icon), 'Teléfono:' 123456, 'Dirección:' Duran, 'Carrera:' Software (dropdown), 'Semestre:' Octavo (dropdown), and 'Materia:' acción y validación de software. At the bottom of the form, there are four navigation buttons: '<|', '<<', '>>', and '>|'.

11. Para Agregar un Conserje nos dirigimos a Secretaría -> nuevo -> Conserje e ingresamos toda la información.

A screenshot of a form titled "Registro de Conserjes". The form contains several input fields and dropdown menus. The fields are: "Cedula:" with the value "12345", "Nombre:" with the value "Angel", "Sexo:" with a dropdown menu showing "Masculino", "Fecha Nac:" with the value "22 mar 2001" and a calendar icon, "Teléfono:" with the value "12345", "Dirección:" with the value "Gye", "Fecha de Ingreso:" with the value "21 jun 2023" and a calendar icon, "Turno:" with a dropdown menu showing "Matutino", "Instrucción:" with the value "Bachiller", and "Sector:" with the value "NA". At the bottom right, there is a "Guardar" button.

12. Para editar los datos nos vamos a secretaria -> editar -> Conserje y se abrirá una ventana para buscar los datos y proceder a editar los datos.

A screenshot of a window titled "Búsqueda del Conserje". It contains a search field labeled "Cedula:" with a text input box. To the right of the input box is a "Buscar" button.



Visualización de Conserjes

Cedula: 12345 Nombre: Angel

Sexo: Masculino Fecha Nac: 22 mar 2001

Teléfono: 12345 Dirección: Gye

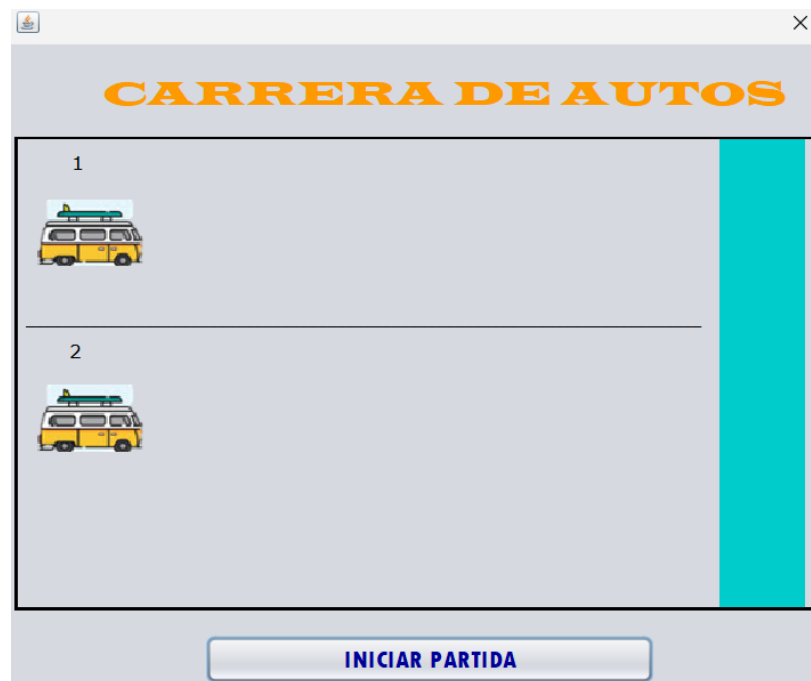
Fecha de Ingreso: 22 mar 2001 Turno:

Instrucción: Bachiller Sector: NA

13. También la ventana principal cuenta con un pequeño juego que se abrirá de la siguiente manera



14. Iniciamos la partida y esperamos un ganador



11.9. Usuarios de sistema operativo

Incluir un listado con usuarios del sistema operativo que utiliza la aplicación, incluyendo:

Nombre del usuario: **Admin.**

El rol del usuario Administrador en un sistema de software tiene como objetivo principal gestionar y supervisar las actividades relacionadas con la administración y configuración del sistema/Gestión de usuarios: El Administrador se encarga de crear, modificar y eliminar cuentas de usuario en el sistema. Esto implica asignar roles y permisos adecuados a cada usuario según sus responsabilidades y necesidades.

Configuración del sistema: El Administrador es responsable de configurar y mantener el sistema de software. Esto puede incluir la instalación de actualizaciones, configuración de parámetros de rendimiento, gestión de licencias, entre otros aspectos técnicos.

Mantenimiento del sistema: El Administrador realiza tareas de mantenimiento regular para asegurar el correcto funcionamiento del sistema. Esto implica realizar copias de seguridad, monitorizar los recursos del sistema, solucionar problemas técnicos y aplicar parches de seguridad.

Seguridad del sistema: El Administrador tiene la responsabilidad de garantizar la seguridad del sistema de software. Esto implica establecer políticas de seguridad, controlar el acceso de usuarios, gestionar certificados digitales, implementar medidas de prevención de intrusiones y realizar auditorías de seguridad.

Supervisión y control: El Administrador supervisa el funcionamiento del sistema y monitoriza su rendimiento. Esto implica identificar posibles problemas, analizar registros de actividad, gestionar el uso de recursos y tomar medidas para optimizar el rendimiento del sistema.

Soporte técnico: El Administrador brinda soporte técnico a los usuarios del sistema, respondiendo a consultas, solucionando problemas y proporcionando asistencia en la resolución de incidencias.

Grupos a los que pertenece: Administrador

Privilegios sobre carpetas: Si

11.10. Contingencias y soluciones.

Contingencias en el programa de gestión de usuarios:

1. Pérdida de datos: Si se produce una pérdida de datos, ya sea por un fallo del sistema o por un error humano, puede afectar la integridad de la información de los usuarios. También podría provocar la incapacidad de acceder a los perfiles de los usuarios o a sus datos personales.
2. Ataques de seguridad: Los programas de gestión de usuarios almacenan información sensible, como contraseñas y datos personales. Existe el riesgo de que se produzcan ataques informáticos, como intentos de robo de información o intrusiones en el sistema, lo que podría comprometer la seguridad de los usuarios.
3. Problemas de rendimiento: Si el programa de gestión de usuarios no está optimizado o no tiene en cuenta un aumento en la cantidad de usuarios, podría enfrentar problemas de rendimiento, como tiempos de respuesta lentos o caídas del sistema, lo que afectaría la experiencia del usuario y la eficiencia del programa.

Soluciones para las contingencias:

1. Implementar copias de seguridad regulares: Realizar copias de seguridad periódicas de la información de los usuarios garantiza que, en caso de pérdida de datos, sea posible restaurar la información a un estado anterior. Estas copias de seguridad deben almacenarse en ubicaciones seguras y probadas regularmente para asegurarse de que los datos se puedan recuperar correctamente.
2. Fortalecer la seguridad: Implementar medidas de seguridad robustas, como autenticación de dos factores, encriptación de datos, políticas de contraseñas seguras y auditorías regulares del sistema, puede ayudar a mitigar los riesgos de ataques de seguridad. También es importante mantener el software actualizado con las últimas correcciones de seguridad.
3. Optimización del rendimiento: Realizar pruebas exhaustivas de rendimiento para identificar posibles cuellos de botella y optimizar el programa de gestión de usuarios. Esto implica ajustar el código, mejorar las consultas a la base de datos y escalar los recursos del sistema según sea necesario. También se pueden implementar técnicas de almacenamiento en caché para acelerar la respuesta del sistema.
4. Plan de contingencia y recuperación ante desastres: Desarrollar un plan de contingencia que detalle los pasos a seguir en caso de una contingencia grave, como un fallo del sistema o una violación de seguridad. Esto puede incluir la designación de un equipo de respuesta a incidentes, la comunicación con los usuarios afectados y la implementación de medidas correctivas.

Conclusiones.

- A través del estudio y aplicación de las técnicas de análisis estático y dinámico en el proceso de Verificación y Validación (V&V), se logró comprender su importancia en la mejora de la calidad del software y la detección temprana de errores.
- La utilización de herramientas específicas como JUnit, Checkstyle y CyVis permitió facilitar y automatizar el proceso de análisis estático y dinámico, proporcionando resultados más precisos y eficientes.
- La aplicación de técnicas de análisis estático permitió identificar y corregir errores de sintaxis, inconsistencias y malas prácticas en el código fuente, lo cual contribuyó a mejorar la calidad y mantenibilidad del software.
- Mediante el análisis dinámico, fue posible evaluar el comportamiento del software en tiempo de ejecución, detectar errores, excepciones y evaluar el rendimiento del programa de administración de una escuela.
- La evaluación de la complejidad del código a través de métricas como la complejidad ciclomática proporcionó información valiosa sobre la estructura y diseño del software, permitiendo identificar áreas que requerían optimización y reducir posibles riesgos.
- La aplicación de estándares y guías de desarrollo, como Java de Google, ayudó a mantener un código más legible, coherente y de mayor calidad, facilitando su comprensión y mantenimiento por parte del equipo de desarrollo.
- La aplicación de técnicas de V&V en el programa de administración de una escuela permitió validar su cumplimiento de requisitos funcionales y no funcionales, garantizando la calidad y confiabilidad del software.
- La evaluación de la efectividad de las técnicas de V&V utilizadas en el proyecto destacó la importancia de contar con un proceso de verificación y validación sólido, y permitió identificar áreas de mejora y oportunidades para optimizar futuros proyectos.

Referencias Bibliográficas.

1. *Introduction*. (s/f). Oracle.com. Recuperado el 20 de junio de 2023, de

<https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html>

Checkstyle – checkstyle 10.12.0. (s/f). Checkstyle.org. Recuperado el 20 de junio de 2023,

de <https://checkstyle.org/>

Google Java Style Guide. (s/f). Github.io. Recuperado el 20 de junio de 2023, de

<https://google.github.io/styleguide/javaguide.html>

Pressman, R., & Maxim, B. (2019). *Software Engineering: A Practitioner's Approach* (9a ed.). McGraw-Hill Education.