



**Universidad
Tecnológica
del Perú**

Avance 2:

“Sistema de Gestión Escolar EduNet”

CURSO:

Algoritmos y estructura de Datos

SECCIÓN:

20429

PROFESOR:

Carlos Efrain Palomino Vidal

INTEGRANTES:

Zarabia Gamboa Armando - U23313736

Cardeña Cusi Adilson Aldair - U22313069

Talledo Ortiz Kevin Samuel - U23300181

Rojas Marcos Sebastián- U23321009

Tejada Ortiz Cristofer Jose – U22203833

Año:

2024

Índice

1. Capítulo 1	5
1.1 Introducción	5
1.2 Aspectos Generales	5
1.3 Organigrama	6
1.4 Misión y Visión	7
1.4.1 Misión	7
1.4.2 Visión	7
1.5 Objetivos Estratégicos	7
1.5.1 Control de Asistencia y Monitoreo	7
1.5.2 Registro de Alumnos y Secciones	8
1.5.3 Gestión de Horarios	9
1.5.4 Optimización de Procesos Administrativos	10
1.6 Problemática	11
1.7 Alternativas de Solución	11
1.7.1 Mantener el método manual de registro	11
1.7.2 Uso de hojas de cálculo	12
1.7.3 Desarrollo de un sistema automatizado local	12
1.7.4 Implementación de un sistema web con base de datos	12
1.8 Solución Elegida	13
1.8.1 Funcionamiento	13
2. Capítulo 2	14
2.1 Estado de Arte	14
2.2 Marco Teórico	15

2.2.1 Estructura de datos.....	16
2.2.2 Listas enlazadas	16
2.2.3 Nodos	17
2.2.4 Listas genéricas	18
2.2.5 Algoritmos	18
2.2.6 Conexión a Base de datos	19
3. Capítulo 3	20
3.1 Alcance	20
3.2 Requerimientos funcionales y no funcionales	21
3.2.1 Administración de Grados y Secciones	21
3.2.2 Administración de Estudiantes	21
3.2.3 Administración de Docentes	22
3.2.4 Registro de Asistencias y Faltas	22
3.2.5 Elaboración de Reportes	22
3.2.6 Facilidad de Uso (Usabilidad)	22
3.2.7 Capacidad de Expansión (Escalabilidad)	23
3.2.8 Desempeño (Rendimiento)	23
3.2.9 Protección y Seguridad	23
3.2.10 Disponibilidad del Sistema	23
3.2.11 Facilidad de Mantenimiento	23
3.3 Restricciones	24
3.3.1 Restricciones Técnicas	24
3.3.2 Restricciones Funcionales	24
3.3.3 Restricciones de Tiempo y Recursos	24
3.3.4 Restricciones de Seguridad y Privacidad	25

3.4 Diagrama de Clases	25
3.5 Prototipo e Interfaces	26
3.5.1 Login	26
3.5.2 Registro Estudiante	27
3.5.3 Profesor Curso	29
3.5.4 Estudiante Asistencia	31
3.5.5 Controlador	36
3.5.6 Vista.....	40
Bibliografía	43

1. CAPITULO 1

1.1 Introducción

El presente proyecto plantea el desarrollo de un sistema académico integral destinado a una institución educativa, cuyo propósito principal es optimizar y centralizar la gestión de la información académica. A través de esta herramienta, el personal podrá administrar de manera organizada los grados, secciones, cursos, horarios y datos de los estudiantes, así como la asignación de docentes a cada aula. Además, el sistema contará con un módulo especializado de control de asistencia, que permitirá registrar y consultar de forma precisa las faltas y asistencias de los alumnos, facilitando el seguimiento de su rendimiento académico.

Una de las características más relevantes de la propuesta es la posibilidad de acceder a información detallada y actualizada de cada estudiante, incluyendo sus datos personales, el grado y la sección en la que se encuentra matriculado, los cursos que lleva, su horario de clases y su historial de asistencia. De igual manera, los docentes podrán visualizar con claridad los cursos que dictan y las secciones a su cargo, mejorando la organización de los recursos educativos y favoreciendo una gestión académica más eficiente.

Este sistema se plantea como respuesta a las dificultades que genera la administración manual y poco estructurada de la información. Al integrar en una sola plataforma todos los procesos mencionados, se busca reducir tiempos de gestión, minimizar errores y fortalecer el control académico y administrativo de la institución.

1.2 Aspectos Generales

Este sistema será implementado en una institución educativa privada ubicada en el distrito de Los Olivos, Lima. Dicha institución provee servicios educativos en los niveles de

inicial, primaria y secundaria. Aunque se cuente con personal capacitado y una buena rubrica centrada en las ciencias, el la institución se enfrenta a diferentes problemas relacionados a la gestión de información por la falta de un modelo automatizado de registro. Esta gestión manual de registros de asistencias, listas y horarios de clase ha resaltado diferentes campos altamente susceptible a errores.

En este proyecto, el colegio busca optimizar el registro y la consulta de las asistencias de los alumnos además de ampliar la funcionalidad del sistema para añadir diferentes opciones como gestión de promedios, secciones, cursos y los horarios. Esto permitirá una administración más rápida y efectiva de los recursos digitales y del seguimiento del aprendizaje de los estudiantes durante su vida escolar.

1.3 Organigrama

Figura 1

Organigrama colegio visionarios 2025



Nota. Elaboración propia.

1.4 Mision y Vision

1.4.1 Misión

Brindar a la institución educativa una herramienta tecnológica innovadora y automatizada que simplifique y optimice la gestión académica en los niveles de inicial, primaria y secundaria. Nuestro propósito es garantizar un registro confiable de asistencias, promedios, horarios y cursos, facilitando el trabajo de los docentes y mejorando la experiencia de aprendizaje de los estudiantes, todo ello mediante un sistema accesible y seguro.

1.4.2 Visión

Convertirnos en un referente en la transformación digital de la gestión educativa en instituciones privadas, destacándonos por implementar soluciones tecnológicas que fortalezcan la eficiencia administrativa, mejoren la comunicación entre docentes, estudiantes y familias, y promuevan un seguimiento académico integral. Aspiramos a contribuir a la formación de estudiantes competentes, con valores sólidos y preparados para enfrentar los desafíos del futuro.

1.5 Objetivos Estratégicos

1.5.1 Control de Asistencia y Monitoreo:

Los sistemas de gestión académica permiten centralizar y automatizar diversos procesos relacionados con la administración educativa. La integración de módulos como el control de asistencia, el registro de alumnos, los datos de los docentes y la organización por secciones y grados es fundamental para optimizar la operación de las instituciones educativas, garantizando un flujo de información eficiente y seguro. Yahya y Anwar (2013)

destacan que la digitalización del registro de asistencia no solo facilita y acelera el proceso, sino que también permite un seguimiento más claro y accesible de la participación académica de los estudiantes. Este tipo de sistemas ofrece a docentes y autoridades una visión detallada y confiable de la asistencia diaria, contribuyendo a una mejor toma de decisiones. En un sistema desarrollado en Java, el control de asistencia puede implementarse como un módulo independiente dentro del menú principal. Dicho módulo permitiría registrar la asistencia diaria, consultar reportes y generar estadísticas de participación. La información se almacenaría inicialmente en memoria mediante estructuras de datos eficientes, para luego sincronizarse con una base de datos que garantice la persistencia y seguridad de los registros.

Estructura de datos. Para gestionar la asistencia se utilizará una LinkedList en Java. Cada elemento de la lista representará un registro de asistencia, almacenando el identificador del estudiante (por ejemplo, su DNI) y su estado (presente, ausente o tardanza) dentro de un objeto. Esta estructura enlazada permite insertar y recorrer registros con facilidad, manteniendo el orden cronológico y facilitando las actualizaciones secuenciales.

1.5.2 Registro de Alumnos y Secciones:

El registro de alumnos y su adecuada organización en grados y secciones constituyen un pilar esencial en la gestión académica, pues garantizan un control eficiente de la información institucional. Según Torres Ramírez (2019), la correcta categorización de los estudiantes por secciones y la asignación de códigos únicos facilita la búsqueda y el manejo de la información de cada alumno. Asimismo, Loyola Cardozo (2019) señala que la incorporación de sistemas informáticos en la gestión educativa no solo agiliza el control de registros, la atención de consultas y la emisión de reportes, sino que también reduce significativamente los tiempos de procesamiento, evidenciando la eficacia de las

herramientas tecnológicas en la administración escolar. El proceso de registro y organización de estudiantes puede implementarse como un módulo del sistema encargado de almacenar datos como DNI, nombre, apellido y número de cursos. Este módulo permitirá registrar nuevos alumnos, actualizar su estado y consultar su información de manera rápida. Además, se integrará con otros procesos, como el control de asistencia o la gestión de cursos, garantizando una base de datos centralizada y accesible.

Estructura de datos. Para manejar a los estudiantes se empleará una clase Alumno y una LinkedList para las secciones. Cada nodo representará los atributos de los alumnos dentro de la lista correspondiente a cada grado o sección. Esta estructura facilita la organización y el acceso a la información, permitiendo un control ordenado y eficiente.

1.5.3 Gestión de horarios:

La gestión de horarios y espacios tanto para el personal educativo como para los estudiantes resulta esencial en instituciones de gran tamaño, donde múltiples cursos se desarrollan simultáneamente. Digital Ware (2025) menciona que la adecuada asignación de recursos y espacios físicos (como aulas, proyectores, mesas, sillas y computadoras) es crucial para garantizar un entorno de aprendizaje óptimo. Cada área cuenta con una capacidad limitada, por lo que los horarios deben registrarse en un sistema capaz de asignar, editar o advertir sobre conflictos de uso. Un sistema automatizado puede detectar cruces de horarios o sobreasignaciones, reduciendo los errores humanos y mejorando la eficiencia organizativa.

Estructura de datos. Para este proceso se utilizará una estructura de Árbol, ya que permite almacenar y jerarquizar distintos tipos de datos —como alumnos, aulas, docentes e intervalos de tiempo—, facilitando la detección de conflictos y la gestión dinámica de horarios.

1.5.4 Optimización de Procesos Administrativos

La digitalización, automatización y gestión eficiente de los recursos son pilares fundamentales para optimizar los procesos administrativos. La digitalización permite centralizar matrículas, calificaciones, asistencias y pagos en una sola plataforma, reduciendo el uso de papel y las posibilidades de errores humanos. La automatización facilita la emisión de boletas, certificados, recordatorios y reportes, ahorrando tiempo en tareas repetitivas. Según Alburquerque-Dávila, Davis-Carrillo y Esteves-Fajardo (2024), “la adopción de tecnologías digitales incide en los procesos administrativos de instituciones educativas, evidenciando mejoras en tiempos, reducción de errores y una mayor eficacia en la gestión”. Una gestión eficiente de recursos también incluye la digitalización de inventarios y el control de asistencia del personal, contribuyendo a la mejora continua del servicio educativo y a la satisfacción tanto del personal docente como de los estudiantes.

Estructura de datos. La implementación del módulo de control de procesos administrativos en Java puede emplear un ArrayList para almacenar y gestionar los registros correspondientes a cada trámite o proceso institucional, como matrículas, pagos o solicitudes. Esta estructura permite agregar, eliminar y actualizar información de manera dinámica, adaptándose al crecimiento del sistema sin requerir un tamaño fijo. Además, su acceso directo por índice facilita la búsqueda y generación de reportes sobre el estado de cada proceso, optimizando la eficiencia en la gestión interna. Gracias a su flexibilidad y simplicidad, el uso de ArrayList contribuye a mantener una administración ordenada, coherente y fácilmente escalable dentro del entorno académico.

1.6 Problemática

En la actualidad, la gestión académica dentro de la institución se lleva a cabo de forma manual, lo que ocasiona diversos inconvenientes que afectan tanto la eficiencia como la calidad del trabajo administrativo. Uno de los principales problemas es el registro manual de asistencias, un procedimiento susceptible a equivocaciones, que demanda demasiado tiempo y dificulta el acceso rápido a la información.

Asimismo, la dispersión de datos entre horarios, cursos, alumnos y docentes complica la integración y el acceso unificado de la información, lo que limita la capacidad de la institución para tomar decisiones adecuadas. Esto no solo repercute en la labor de docentes y personal administrativo, sino también en la experiencia de los estudiantes, quienes carecen de un sistema que facilite un seguimiento oportuno y exacto de su rendimiento académico.

En consecuencia, se plantea la necesidad de implementar un sistema de gestión académica que concentre la información, reduzca errores en los registros y automatice tareas esenciales, optimizando la eficiencia administrativa y el proceso de toma de decisiones.

1.7 Alternativas de Solución

1.7.1 Mantener el método manual de registro:

Consiste en continuar utilizando hojas impresas para el control de asistencias, listas de alumnos y horarios.

- *Ventajas:* No requiere inversión tecnológica inicial.
- *Desventajas:* Presenta alta probabilidad de errores, pérdida de información y dificultad para acceder rápidamente a los registros.

1.7.2 Uso de hojas de cálculo:

Implementar formatos digitales en herramientas como Excel para el registro de asistencias, notas y horarios.

- *Ventajas:* Bajo costo, fácil implementación y familiaridad con la herramienta por parte de los docentes.
- *Desventajas:* Poca escalabilidad, dificultades en la gestión de grandes volúmenes de información y riesgo de inconsistencias si varios usuarios modifican los archivos.

1.7.3 Desarrollo de un sistema automatizado local:

Diseñar e implementar un software en lenguaje Java que permita el registro de alumnos, control de asistencias, gestión de notas y administración de horarios.

- *Ventajas:* Disminuye errores de registro, agiliza las consultas y permite personalizar las funcionalidades según las necesidades del colegio.
- *Desventajas:* Requiere capacitación del personal y se limita a equipos donde esté instalado.

1.7.4 Implementación de un sistema web con base de datos en la nube:

Crear una plataforma en línea a la que puedan acceder directivos y docentes desde cualquier dispositivo con conexión a internet.

- *Ventajas:* Acceso remoto, centralización de la información y mayor escalabilidad.
- *Desventajas:* Demanda una mayor inversión en infraestructura y dependencia de la conectividad a internet.

1.8 Solución Elegida.

Tras analizar las alternativas, se opta por la tercera propuesta: el desarrollo de un sistema automatizado local en lenguaje Java. Esta elección se fundamenta en que brinda un equilibrio entre costo, funcionalidad y facilidad de implementación, permitiendo una transición progresiva desde el sistema manual hacia uno digitalizado.

1.8.1 Funcionamiento

- **Arreglos:** Para almacenar y gestionar listas de alumnos en cada curso.
- **Matrices:** Para organizar los horarios de clase, estructurados por días y horas.
- **Listas doblemente enlazadas:** Para listas la asistencia de los alumnos.
- **Algoritmos de ordenamiento (selección y burbuja):** Para organizar registros de alumnos por apellido o asistencia.
- **Algoritmos de ordenamiento avanzados (merge sort y quick sort):** Para ordenar promedios o calificaciones de forma eficiente en grandes volúmenes de datos.
- **Búsqueda secuencial y binaria:** Para localizar de manera rápida a un estudiante según su nombre o código.

2. CAPITULO 2

2.1 Estado de arte.

La transición de la gestión académica manual a la digital ha generado un campo de estudio significativo, especialmente en la implementación de sistemas de control de asistencia y administración escolar. La revisión de la literatura reciente evidencia una clara tendencia hacia la automatización como solución a las ineficiencias operativas.

Investigaciones previas, como la de Torres Ramírez (2019), ya señalaban la insatisfacción del personal (un en su estudio) con los métodos de control de asistencia manual, proponiendo soluciones tecnológicas como el código QR para mejorar la precisión y agilizar el registro de docentes y administrativos. Esto establece la necesidad fundamental de migrar de lo manual a lo digital.

La pertinencia de estos sistemas se extiende incluso a contextos extraordinarios. Rubio Ortiz (2022) demostró, a través de un desarrollo bajo la metodología Scrum, la capacidad de estas plataformas para adaptarse a la virtualidad (post-COVID-19), creando un registro de asistencias sincrónicas. Su trabajo subraya la importancia de generar reportes en tiempo real para mantener informados tanto a docentes como a padres de familia, reforzando la necesidad de una herramienta de seguimiento constante.

Desde una perspectiva más amplia de gestión, el desarrollo de sistemas integrales es crucial. El trabajo de Figueroa Dumes y Macías Armendariz (2020) se enfocó en agilizar no solo la asistencia, sino también la evaluación en una plataforma web. Su conclusión sobre la optimización del tiempo y la mejora de la precisión al reemplazar registros obsoletos justifica la inversión en sistemas automatizados.

Finalmente, la relación directa entre el software y la mejora de los resultados académicos está científicamente respaldada. Palomino Ramos (2019) estableció una correlación positiva y significativa () entre el uso de software educativo y la mejora del control académico. Este dato cuantitativo es un argumento sólido a favor del proyecto EduNet, ya que la automatización no solo simplifica la administración, sino que impacta positivamente en el control y seguimiento del rendimiento estudiantil.

2.2 Marco Teórico.

La incorporación de las Tecnologías de la Información y la Comunicación (TIC) ha transformado profundamente la gestión administrativa en las instituciones educativas, especialmente en los colegios. Según Cabero y Llorente (2019), las TIC no solo han modificado los procesos pedagógicos, sino también las formas en que las organizaciones educativas gestionan su información, coordinan sus recursos y establecen comunicación con la comunidad escolar.

En este contexto, la automatización administrativa se entiende como la aplicación de sistemas computacionales capaces de ejecutar tareas de gestión de manera autónoma o semiautónoma, con el fin de optimizar el uso de los recursos humanos y materiales. Esta automatización se refleja en la digitalización de registros de matrícula, control de asistencia, generación de reportes, y en la integración de bases de datos académicas y administrativas (García & Paredes, 2021).

La adopción de plataformas de gestión escolar, como Siagie, EducaNet, School Manager o sistemas ERP educativos, ha permitido a los colegios disponer de información centralizada, reducir errores humanos y agilizar la toma de decisiones. Según Osorio Giraldo, Ramírez y Rojas (2020), estos sistemas fortalecen la comunicación institucional y aumentan

la eficiencia operativa al automatizar procedimientos repetitivos y estandarizar los flujos de trabajo.

Desde una perspectiva organizacional, la automatización impulsa la transición de modelos burocráticos tradicionales hacia modelos de gestión inteligente, en los cuales los procesos se orientan a resultados y se sustentan en el análisis de datos. Esto no solo contribuye a una administración más eficiente, sino que también promueve la transparencia, la rendición de cuentas y la mejora continua de la calidad educativa.

Por su parte, la UNESCO (2021) destaca que la transformación digital y la automatización administrativa constituyen pilares fundamentales para el fortalecimiento de sistemas educativos más resilientes, inclusivos y sostenibles. La adopción de estas herramientas tecnológicas permite que las instituciones educativas respondan con mayor agilidad a los cambios sociales, tecnológicos y normativos del entorno.

En síntesis, la automatización de los procesos administrativos en los colegios representa una oportunidad estratégica para mejorar la gestión escolar. Al integrar la tecnología con la administración educativa, se optimizan los recursos, se incrementa la eficiencia institucional y se sientan las bases para una educación de calidad adaptada a las demandas del presente siglo.

2.2.1 Estructuras de Datos

Las estructuras de datos son formas organizadas de almacenar y administrar la información dentro de un programa, permitiendo su manipulación eficiente. Su elección influye directamente en el rendimiento de los algoritmos y en la claridad del código. Entre las más comunes se encuentran las listas, pilas, colas, árboles y grafos. En el caso del presente sistema, se utilizan principalmente listas enlazadas simples y dobles, que proporcionan flexibilidad en la inserción y eliminación de elementos.

2.2.2 Listas Enlazadas

Una lista enlazada es una colección de nodos conectados entre sí mediante referencias o punteros. Cada nodo contiene un dato y una referencia al siguiente nodo (o al anterior, en el caso de listas dobles).

Existen varios tipos de listas enlazadas:

- **Lista enlazada simple:** cada nodo apunta únicamente al siguiente nodo. Su recorrido es unidireccional. Este tipo de lista es ideal para operaciones secuenciales y dinámicas, como la utilizada en la clase `ListaEnlazadaSimple<T>`.
- **Lista doblemente enlazada:** cada nodo mantiene referencias al nodo anterior y al siguiente, permitiendo recorrer la estructura en ambas direcciones. En el proyecto, las clases `ListaAsistencia` y `ListaAsistenciaMeses` utilizan este tipo de estructura para manejar los registros de asistencia y los meses de forma bidireccional.

Las operaciones básicas que se aplican sobre las listas enlazadas incluyen:

- **Inserción:** agregar un nodo al inicio o al final de la lista.
- **Eliminación:** retirar un nodo específico manteniendo la coherencia de los enlaces.
- **Búsqueda:** recorrer secuencialmente los nodos para encontrar un elemento.
- **Recorrido:** procesar cada nodo desde el inicio hasta el final (o viceversa).

Estas estructuras son útiles cuando se requiere un manejo dinámico de los datos, ya que no es necesario definir un tamaño fijo como en los arreglos.

2.2.3 Nodos

El nodo es la unidad fundamental de las listas enlazadas. Contiene el valor almacenado y las referencias necesarias para enlazarse con otros nodos. En la clase `Nodo<T>` se implementa de manera genérica, permitiendo almacenar cualquier tipo de objeto.

El diseño genérico del nodo permite reutilizar la estructura para diferentes propósitos, como listas de cursos, profesores o asistencias.

2.2.4 Listas Genéricas

El uso de parámetros genéricos ($\langle T \rangle$) permite crear estructuras de datos reutilizables sin importar el tipo de información que contengan. Gracias a esta característica, la clase `ListaEnlazadaSimple<T>` puede manipular listas de profesores, cursos o cualquier otro tipo de objeto. Este enfoque incrementa la abstracción y promueve la reutilización del código, un principio clave en la programación orientada a objetos.

2.2.5 Algoritmos

Un algoritmo es un conjunto finito de pasos ordenados que permiten resolver un problema o realizar una tarea. En estructuras enlazadas, los algoritmos se implementan para recorrer, insertar, eliminar o buscar elementos.

La eficiencia de estos algoritmos se evalúa comúnmente en términos de complejidad temporal y espacial.

- **Algoritmos de Inserción:** El proceso de inserción en una lista enlazada implica crear un nuevo nodo y ajustar las referencias de los nodos adyacentes. En una lista simple, la inserción al final requiere recorrer toda la lista hasta el último nodo. En una lista doble, la inserción puede hacerse directamente si se mantiene una referencia al nodo final, reduciendo el tiempo de ejecución.
- **Algoritmos de Eliminación:** Eliminar un nodo requiere localizar el elemento y modificar los punteros de sus vecinos. En las clases `ListaAsistencia` y `ListaAsistenciaMeses`, este proceso incluye casos especiales para cuando el nodo a eliminar se encuentra al inicio o al final de la lista.

- **Algoritmos de Búsqueda:** La búsqueda en una lista enlazada se realiza de forma secuencial, comparando elemento por elemento hasta hallar el valor deseado o llegar al final. Aunque este método tiene una complejidad de $O(n)$, su implementación es sencilla y suficiente para volúmenes de datos moderados.
- **Recorridos Bidireccionales:** En listas doblemente enlazadas, los recorridos pueden realizarse tanto de izquierda a derecha como de derecha a izquierda. Este enfoque se observa en métodos como `listarIzquierdaDerecha()` y `listarDerechaIzquierda()`, que facilitan la visualización completa de los registros desde cualquier extremo.

2.2.6 Conexión con Bases de Datos

Aunque no se considera una estructura de datos en memoria, la conexión a base de datos mediante la clase `ConexionBD` representa una estructura lógica para la persistencia de información.

Esta conexión utiliza el modelo Cliente-Servidor con el protocolo JDBC, permitiendo la comunicación entre la aplicación Java y un servidor MySQL. Los algoritmos implicados se centran en la ejecución de consultas SQL y la gestión de recursos mediante la clase `LoginDAO`, que aplica validaciones mediante consultas parametrizadas para evitar inyecciones SQL.

3. CAPITULO 3

3.1 Alcance:

El sistema de gestión desarrollado tiene como propósito optimizar la administración educativa dentro de una institución escolar. Su función principal será centralizar la información académica y administrativa, integrando datos de estudiantes, docentes, cursos, secciones, horarios y asistencia en una sola plataforma. De esta manera, se busca lograr una gestión más eficiente, organizada y accesible para los responsables de la administración educativa.

El sistema permitirá:

- Administrar grados, secciones y cursos: Se podrán registrar y organizar los distintos niveles educativos (como Primaria o Secundaria), cada uno dividido en secciones con un código único de identificación. Asimismo, será posible asignar los cursos correspondientes a cada grado y sección.
- Gestionar alumnos: El sistema facilitará el registro de estudiantes en las secciones correspondientes, almacenando información personal (nombres, apellidos, grado, sección y cursos matriculados). Además, mostrará los horarios y el historial de asistencia de cada alumno.
- Gestionar docentes: Se incluirá un módulo para registrar nuevos profesores, asignándoles horarios específicos y credenciales de acceso personalizadas con permisos según su rol.
- Registrar asistencia: Cada estudiante contará con un historial detallado de asistencia, donde se reflejarán sus asistencias y faltas, lo que permitirá realizar un seguimiento preciso a lo largo del semestre académico.

- Generar reportes: El sistema podrá emitir reportes detallados sobre la asistencia de los alumnos, accesibles tanto para los administradores como para los docentes.

Este sistema se desarrollará como una plataforma web, disponible desde cualquier dispositivo conectado a la red institucional, lo que garantizará un acceso ágil y seguro a la información académica.

3.2 Requerimientos funcionales y no funcionales:

Los requerimientos funcionales describen las principales acciones que el sistema debe ejecutar para cumplir adecuadamente con las expectativas y necesidades de los usuarios.

3.2.1 Administración de Grados y Secciones:

- El sistema deberá ofrecer la posibilidad de registrar nuevos grados, así como modificar o eliminar los ya existentes.
- Permitirá crear y actualizar secciones, asignando un identificador único a cada una de ellas.
- Deberá permitir la vinculación de los cursos a las secciones correspondientes, junto con la planificación de sus horarios.

3.2.2 Administración de Estudiantes:

- El sistema registrará la información personal del alumno, incluyendo nombres, apellidos, DNI, datos del apoderado, grado, sección y los cursos que tiene asignados.
- Mostrará el horario de clases correspondiente a cada estudiante.
- Permitirá generar y consultar el historial de asistencia, mostrando días de inasistencia y tardanza, además de ofrecer la opción de exportar reportes en formato PDF.

3.2.3 Administración de Docentes:

- El sistema almacenará los datos personales de los profesores, como nombres, apellidos, DNI, usuario y contraseña.
- Mostrará los horarios de clases que cada docente tiene asignados.

3.2.4 Registro de Asistencias y Faltas:

- Permitirá que el responsable del control registre las inasistencias y tardanzas de los estudiantes.
- Generará reportes de faltas clasificados por alumno, curso, sección o docente.
- Ofrecerá la posibilidad de consultar registros de asistencia dentro de rangos de fechas específicos.

3.2.5 Elaboración de Reportes:

- El sistema generará de manera automática informes personalizables sobre asistencia, horarios y distribución de docentes.
- Los requerimientos no funcionales detallan las propiedades de calidad que debe poseer el sistema para garantizar un funcionamiento eficiente, seguro y estable.

3.2.6 Facilidad de Uso (Usabilidad):

- El sistema contará con una interfaz gráfica clara y sencilla, de modo que pueda ser utilizada sin dificultad por personal sin formación técnica.
- Se brindarán breves sesiones de capacitación para que los usuarios comprendan y apliquen correctamente las funciones principales.

3.2.7 Capacidad de Expansión (Escalabilidad):

- El sistema debe poder adaptarse al crecimiento del número de usuarios, tanto docentes como alumnos, sin que esto afecte su rendimiento.
- Asimismo, debe permitir incorporar nuevas funcionalidades o módulos adicionales en el futuro.

3.2.8 Desempeño (Rendimiento):

- El sistema deberá registrar y procesar la asistencia, además de generar reportes de manera inmediata, con un margen máximo de retraso de 5 segundos.
- Las consultas de información deberán responder en un tiempo no superior a 3 segundos por solicitud.

3.2.9 Protección y Seguridad:

- Se implementarán mecanismos de autenticación y control de acceso para garantizar que solo los usuarios autorizados (como administradores o docentes) puedan visualizar datos sensibles.
- Los datos personales de alumnos y profesores deberán mantenerse protegidos mediante técnicas de cifrado.

3.2.10 Disponibilidad del Sistema:

- El sistema deberá permanecer operativo al menos el 99% del tiempo, reduciendo al mínimo las interrupciones durante el horario académico.

3.2.11 Facilidad de Mantenimiento:

- El software debe ser de fácil mantenimiento, permitiendo realizar actualizaciones, correcciones o mejoras sin interrumpir significativamente las actividades cotidianas.

3.3 Restricciones

3.3.1 Restricciones Técnicas:

- El sistema será desarrollado completamente con el lenguaje de programación Java
- La base de datos utilizada será MySQL, conectada mediante JDBC.
- El sistema solo tendrá el requisito de poder correr en el sistema operativo Windows.
- El programa estará limitado a funcionalidad en una red privada o en una sola PC.
- Este trabajo no contemplará diferentes dispositivos como móviles o web.

3.3.2 Restricciones Funcionales:

- El programa no podrá
- El sistema no podrá registrar asistencias sin aula o profesor asignado previamente.
- Cada profesor podrá estar asignado a un máximo de un aula por franja horaria.
- No se permitirá modificar registros de asistencia una vez cerrada la jornada escolar.
- Las aulas deberán estar previamente registradas y validadas por el administrador.

3.3.3 Restricciones de Tiempo y Recursos:

- El desarrollo del sistema deberá completarse en un plazo máximo de 8 semanas.
- El equipo contará con dos programadores, un analista y un encargado de pruebas.
- El proyecto deberá desarrollarse utilizando herramientas de código abierto o con licencias gratuitas.

3.3.4 Restricciones de Seguridad y Privacidad:

- Los datos personales de alumnos y profesores deberán almacenarse cumpliendo con las normas básicas de protección de datos (Ley N° 29733 o similar).
- No se permitirá el acceso simultáneo con las mismas credenciales desde diferentes dispositivos.

La información de asistencia solo podrá ser consultada por usuarios autorizados.

3.4 Diagrama de Clases:

Figura 2

Diagrama de clases del sistema de administración escolar



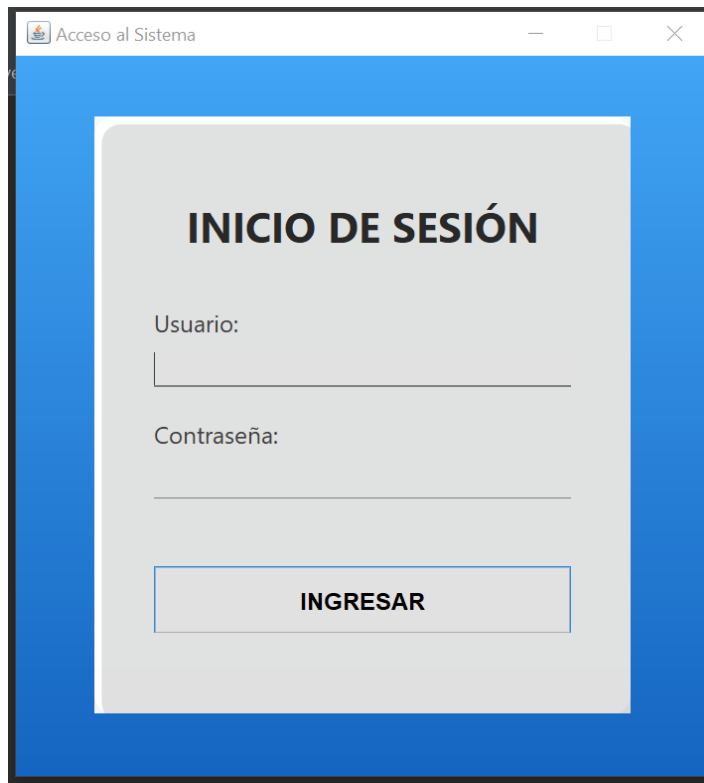
Nota. Elaboración propia. Diagrama de clases diseñado en PlantUML (2025).

3.5 Prototipo y interfaces:

3.5.1 Login

Figura 3

Interfaz de inicio de sesión del sistema de administración escolar



The image shows a Java Swing window titled "Acceso al Sistema". The window has a blue border and a light gray background. In the center, there is a white rectangular area containing the text "INICIO DE SESIÓN" in bold. Below this, there are two input fields: "Usuario:" and "Contraseña:". The "Usuario:" field has a text input line, and the "Contraseña:" field has a password input line (indicated by a horizontal line). Below the input fields, there is a button labeled "INGRESAR".

Nota. Elaboración propia. Interfaz desarrollada en Java Swing (2025).

3.5.2 Registro Estudiante

Figura 4

Interfaz del módulo de registro de estudiantes

Sistema de Gestión Académica Integrado

1. Estudiantes (Ordenamiento) 2. Profesores y Cursos 3. Control de Asistencia

DNI: Nombre: Apellido: Asistencias: Cursos:

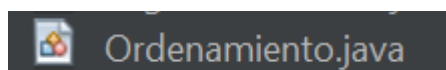
Criterio: Método:

DNI	Nombre	Apellido	Asistencias	Cursos
10000000	Maria	Ramirez	56	3
10000001	Sofia	Lopez	62	3
10000002	Carlos	Perez	57	4
10000003	Lucia	Lopez	95	4
10000004	Diego	Perez	63	4
10000005	Ana	Torres	76	1
10000006	Sofia	Perez	54	3
10000007	Maria	Garcia	78	3
10000008	Lucia	Perez	88	3
10000009	Ana	Garcia	80	4
10000010	Ana	Sanchez	71	3
10000011	Carlos	Gomez	79	3
10000012	Javier	Sanchez	52	2
10000013	Pedro	Gomez	75	4
10000014	Diego	Lopez	67	5
10000015	Carlos	Sanchez	100	2
10000016	Carlos	Gomez	82	2
10000017	Ana	Rodriguez	57	5
10000018	Sofia	Sanchez	62	4
10000019	Carlos	Perez	81	1
10000020	Carlos	Garcia	69	4
10000021	Diego	Torres	87	4

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en Java Swing (2025).

Figura 5

Clase Ordenamiento del sistema de Registro estudiante



Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en lenguaje Java (2025).

- **Clase Ordenamiento:**

La clase Ordenamiento también pertenece al paquete Controlador y tiene como finalidad proveer diferentes algoritmos de ordenamiento aplicables a los objetos de tipo

Estudiante.

No forma parte directa del flujo de interacción entre la vista y el modelo, sino que sirve como una clase de utilidad lógica dentro del sistema académico.

Esta clase implementa varios métodos estáticos que permiten ordenar y registrar arreglos de estudiantes según distintos criterios, como el DNI, nombre, apellido, número de asistencias o cantidad de cursos matriculados.

Los algoritmos implementados son:

- `bubbleSort()` : Ordenamiento burbuja, compara elementos adyacentes y los intercambia si están en el orden incorrecto.
- `selectionSort()` : Ordenamiento por selección, busca el elemento menor y lo coloca en la posición correspondiente.
- `quickSort()` : Ordenamiento rápido, utiliza el método de partición y recursividad para dividir y ordenar el arreglo.
- `mergeSort()` : Ordenamiento por mezcla, divide el arreglo en mitades, las ordena y luego las fusiona en orden.

Cada algoritmo puede ordenar los datos según un criterio dinámico que el usuario o el sistema define en tiempo de ejecución (por ejemplo, “nombre”, “dni” o “asistencias”).

La clase Ordenamiento proporciona una colección modular de algoritmos de clasificación, que fortalecen la lógica interna del sistema y permiten mostrar la información de los estudiantes de manera flexible y organizada.

3.5.3 Profesor Curso

Figura 6

Interfaz del módulo de Profesor Curso

Sistema de Gestión Académica Integrado

1. Estudiantes (Ordenamiento) 2. Profesores y Cursos 3. Control de Asistencia

Datos del Profesor y Curso

ID Profesor:

Nombre Profesor:

Código Curso:

Nombre Curso:

Créditos:

Agregar Curso

Guardar Profesor Buscar Profesor Eliminar Profesor Mostrar Todos

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en Java Swing (2025).

Figura 7

Interfaz del módulo de Profesor Curso con contenido

Sistema de Gestión Académica Integrado

1. Estudiantes (Ordenamiento) 2. Profesores y Cursos 3. Control de Asistencia

Datos del Profesor y Curso

ID Profesor:

Nombre Profesor:

Código Curso:

Nombre Curso:

Créditos:

Agregar Curso

Guardar Profesor Buscar Profesor Eliminar Profesor Mostrar Todos

Alcantara (ID: 124124)
Cursos asignados:
- Algoritmos (123124)

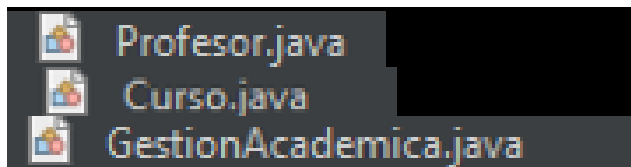
Perez (ID: 31353)
Cursos asignados:
- Estadística (35235)

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en Java Swing (2025).

En las relaciones de Profesor/Curso hemos utilizado diferentes clases para poder realizar un proceso que nos ayudara a guardar una lista de profesores y los cursos que serian asignados a cada uno de ellos.

Figura 8

Clases implementadas al módulo Profesor Curso



Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en lenguaje Java (2025).

En la clase profesor hemos utilizado un constructor y los métodos necesarios para hacer una creación de un objeto. (Constructor Profesor, get/sets y toString)

En la clase Curso utilizamos los mismos métodos que en la clase Profesor pero la diferencia de este es que hemos utilizado un método llamado “equals” que nos servirá para comparar el objeto si es que es una instancia de la clase curso.

Y para la clase GestionAcademica hemos utilizado métodos comunes en para un objeto de lista enlaza como:

- EliminarDocente: Llama al método buscarDocente que confirma y elimina el objeto especificado en caso exista.
- BuscarDocente: Compara los elementos de la lista de forma recursiva hasta que se encuentre el objetivo o hasta que el siguiente objeto sea nulo.

- **AsignarCursoAProfesor:** Se le vincula un objeto “Curso” especificado a un objeto “Profesor” luego de buscarlos utilizando buscarDocente. En caso uno de los valores ingresados sea nulo este método no se realizara.
- **AgregarProfesor:** Se agregara un objeto de la clase profesor a la lista al final de la lista mientras que esta no sea nula, en caso si sea nula este objeto se agregara en la primera y única posición.

3.5.4 Estudiante Asistencia

Figura 9

Interfaz del módulo de Estudiante Asistencia

Sistema de Gestión Académica Integrado

1. Estudiantes (Ordenamiento) 2. Profesores y Cursos 3. Control de Asistencia

Registro Listado

Mes:

DNI:

Nombre:

Apellido:

Asistencias:

Cursos:

Estado: Presente

Agregar al Inicio Agregar al Final

Acciones de Registro

DNI (Eliminar/Verificar):

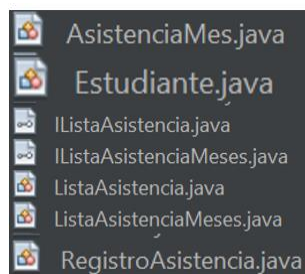
Mes (Eliminar/Verificar):

Eliminar Asistencia Verificar Asistencia Contar Registros

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en Java Swing (2025).

Figura 10*Segunda Interfaz del módulo de Estudiante Asistencia*

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en Java Swing (2025).

Figura 11*Clases implementadas al módulo Estudiante Asistencia*

Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en lenguaje Java (2025).

- **Clase AsistenciaMes:**

La clase AsistenciaMes Contiene los atributos de nombre para los nodos que representaran los meses, anterior y siguiente para poder avanzar o retroceder de nodo dentro de la lista enlazada. Además, contiene un constructor que recibirá como parámetro el nombre del mes y donde se marca la referencia anterior y siguiente como nulo. Por último, la clase contiene los respectivos getters y setters de sus atributos.

- **Clase Estudiante:**

La clase Estudiante pertenece al paquete Modelo y representa la estructura de datos que modela la información de un estudiante dentro de un sistema académico. Forma parte del patrón de arquitectura MVC (Modelo–Vista–Controlador), donde el modelo se encarga de manejar los datos y la lógica interna.

Esta clase permite almacenar, acceder y mostrar la información personal y académica de cada estudiante, como su DNI, nombre, apellido, asistencias y número de cursos matriculados.

- **Interfaz IListaAsistencia:**

La interfaz IListaAsistencia contiene los métodos que se implementaran dentro de la clase ListaAsistencia, donde esta modificara los métodos de acorde a su lógica.

- **Interfaz IListaAsistenciaMeses:**

Así como IListaAsistencia, la interfaz IListaAsistenciaMeses contiene los métodos que serán implementados dentro de la clase ListaAsistenciaMeses, para que esta clase sobrescriba los métodos acordes a su lógica.

- **Clase ListaAsistencia:**

La clase ListaAsistencia implementa la interfaz IListaAsistencia, dentro de su estructura contiene dos atributos de la clase RegistroAsistencia, los cuales son instanciados como nulo dentro de su constructor, además de contener los getters de los mismos.

El método agregarRegistroFinal() verifica si la lista está vacía, en caso de estarlo asigna el nuevo registro como el inicio y el final de la lista, caso contrario es asignado a la posición siguiente del último nodo.

El método agregarRegistroInicio() hace lo mismo que el método agregarRegistroFinal(), con la diferencia de que este método asigna el nuevo nodo al inicio de la lista.

El método eliminarAsistencia() primero verifica si la lista está vacía, si lo está retorna falso y termina. Si la lista contiene elementos, crea un nodo y se le asigna el valor inicial, para que sea recorrido dentro del bucle while, el cual terminará cuando este nodo sea nulo, dentro del bucle se comparará el valor que se desea eliminar verificando nodo por nodo, si encuentra coincidencias el método retorna verdadero y termina. Si el bucle se termina de recorrer y no se encontraron coincidencias el método retorna falso,

El método existeAsistencia() sigue la misma lógica que el método eliminarAsistencia() para encontrar coincidencias, si se encuentra la asistencia, el método termina y retorna verdadero, en caso de que el método termine y no encuentre coincidencias el método terminará y retorna falso.

El método contarRegistros() utiliza la misma lógica que el método existeAsistencia(), con la diferencia de que el bucle recorrerá toda la lista e irá contando cada iteración del bucle para que una vez termine retorne ese valor.

El método `listarIzquierdaDerecha()` primero verifica si la lista esta vacia, si lo esta devuelve el mensaje "Sin registros". De lo contrario se creará un `StringBuilder` para poder imprimir la lista y un nodo al cual se le asignará el valor inicial. Dentro de un bucle `while` en cada iteración se guardara el método `toString()` en el `StringBuilder` para ser impreso.

El método `listarDerechaIzquierda()` hace lo mismo que el método `listarIzquierdaDerecha()` con la diferencia que el nuevo nodo se le asignara el ultimo valor de la lista y usara el método `getAnterior()` para recorrer e imprimir la lista en sentido contrario.

El método `estaVacía()` verifica si la lista esta vacía y retorna verdadero o falso si la lista esta vacía o no.

- **Clase `ListaAsistenciaMeses`:**

El método `agregarMesInicio()` verifica si la lista esta vacía, en caso de estarlo asigna el nuevo mes como el inicio de la lista. Caso contrario es asignado como la cabeza de la lista y su puntero apuntara a la antigua cabeza de la lista.

El método `agregarMesFinal()` verifica si la lista está vacía, en caso de estarlo asigna el nuevo mes al inicio de la lista, caso contrario será asignado en el último puntero del nodo de la lista.

El método `eliminarMes()` primero verifica si la lista está vacía, el método termina. Si la lista contiene elementos, crea un nodo y se le asigna el valor inicial, para que sea recorrido dentro del bucle `while`, el cual terminara cuando este nodo sea nulo, dentro del bucle se comparara el valor que se desea eliminar verificando nodo por nodo, si encuentra coincidencias el nodo será omitido, haciendo que su anterior nodo apunte al siguiente dos veces, de esta forma el nodo es eliminado de la lista.

El método `mostrarMesIzqADer()` primero verifica si la lista esta vacía, si lo está devuelve un mensaje y termina el método. De lo contrario se crea un nodo al cual se le asignará el valor inicial. Dentro de un bucle `while` en cada iteración se imprimirá el mes con la asistencia respectiva.

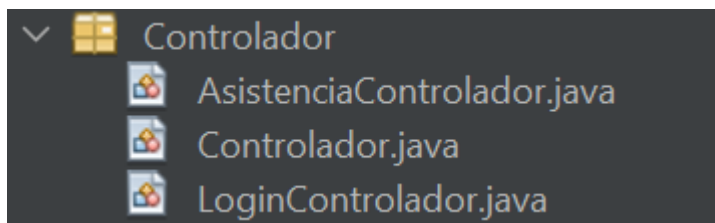
El método `mostrarMesDerAIzq()` hace lo mismo que el método `mostrarMesIzqADer()` con la diferencia que el nuevo nodo se le asignara el valor final y usara el método `getAnterior()` para recorrer e imprimir la lista en sentido contrario.

El método `estaVacia()` verifica si la lista está vacía y retorna verdadero o falso si la lista está vacía o no.

3.5.5 Controlador

Figura 12

Clases implementadas al paquete controlador



Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en lenguaje Java (2025).

- **Clase AsistenciaControlador:**

La clase `AsistenciaControlador` pertenece al paquete `Controlador` y forma parte del patrón de arquitectura MVC (Modelo–Vista–Controlador).

Su función principal es gestionar la comunicación entre la vista y el modelo, actuando como intermediario entre los datos de asistencia y la interfaz que los muestra.

Esta clase utiliza la clase ListaAsistencia del paquete Modelo, la cual contiene la estructura de datos donde se almacenan los registros de asistencia de los estudiantes.

Dentro del constructor, se inicializa una lista de asistencia con algunos registros de ejemplo. Además, el controlador proporciona métodos que permiten:

- Agregar registros al inicio o al final de la lista.
 - Eliminar registros según el DNI y el mes del estudiante.
 - Verificar si un registro existe dentro de la lista.
 - Contar la cantidad total de registros almacenados.
 - Mostrar los registros en orden de izquierda a derecha o de derecha a izquierda, según se requiera.
- **Clase Controlador:**

La clase Controlador pertenece al paquete Controlador y forma parte del patrón de arquitectura MVC (Modelo–Vista–Controlador).

Su principal función es gestionar la comunicación entre la Vista (VentanaPrincipal) y el Modelo (GestionAcademica, Profesor y Curso), coordinando las acciones del usuario con las operaciones del sistema.

Dentro de esta clase se implementa la interfaz ActionListener, lo que permite capturar y procesar los eventos generados por los botones de la interfaz gráfica. De esta manera, cada acción del usuario desencadena una operación específica en el modelo.

El controlador posee los siguientes atributos principales:

- GestionAcademica sistema: representa el modelo principal encargado de almacenar y administrar los profesores y cursos.

- **VentanaPrincipal** vista: hace referencia a la interfaz gráfica que interactúa con el usuario.
- **Profesor profesorActual**: mantiene una referencia temporal al profesor que está siendo gestionado (creado, modificado o consultado).

Las funciones más importantes del controlador son:

- **agregarCurso()**: permite asociar un nuevo curso al profesor actual, validando los datos ingresados por el usuario.
- **guardarProfesor()**: almacena el profesor y sus cursos en el sistema, asegurando que no exista duplicidad de ID.
- **buscarProfesor()**: busca un profesor por su identificador y muestra su información en la vista.
- **eliminarProfesor()**: elimina un profesor del sistema a partir de su ID.
- **mostrarTodo()**: muestra todos los profesores registrados en el sistema.

La clase **Controlador** actúa como el enlace funcional entre la interfaz gráfica y la lógica del programa, garantizando la correcta ejecución de las operaciones académicas y el flujo coherente de datos dentro de la aplicación.

- **Clase LoginControlador:**

La clase **LoginControlador** pertenece al paquete **Controlador** y forma parte del patrón MVC (Modelo–Vista–Controlador).

Su función principal es gestionar la lógica del proceso de inicio de sesión, actuando como intermediario entre la interfaz gráfica de usuario (Vista) y la capa de acceso a datos (Modelo).

Esta clase implementa la interfaz ActionListener, lo que le permite detectar y responder a las acciones del usuario dentro de la vista de inicio de sesión.

Sus componentes principales son:

- LoginVista vista: representa la interfaz donde el usuario ingresa su nombre de usuario y contraseña.
- LoginDAO modeloDAO: es el objeto del modelo encargado de validar las credenciales en la base de datos o conjunto de datos definido.

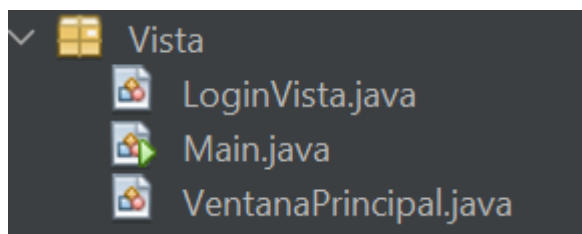
El flujo de trabajo de la clase es el siguiente:

- Captura los datos ingresados por el usuario (usuario y contraseña).
- Valida los campos para evitar entradas vacías.
- Verifica las credenciales a través del método verificarCredenciales() del modelo DAO.
- Si los datos son correctos, abre la ventana principal del sistema (VentanaPrincipal) y cierra la vista de login.
- Si las credenciales son incorrectas, muestra un mensaje de error informando al usuario.

3.5.6 Vista

Figura 13

Clases implementadas al paquete Vista



Nota. Elaboración propia. Captura del entorno NetBeans, desarrollada en lenguaje Java (2025).

- **Clase LoginVista:**

La clase LoginVista pertenece al paquete Vista y representa la interfaz gráfica de usuario (GUI) encargada del proceso de inicio de sesión dentro del sistema académico. Forma parte del patrón de arquitectura MVC (Modelo–Vista–Controlador), donde su papel principal es mostrar la información y capturar las acciones del usuario, sin incluir lógica de negocio.

Esta clase extiende JFrame, por lo que constituye una ventana principal de tipo Swing, diseñada con un estilo moderno y profesional, utilizando componentes personalizados, colores gradientes y bordes redondeados.

Entre sus elementos visuales más importantes se encuentran:

- Campos de texto para ingresar el nombre de usuario (txtUsuario) y la contraseña (txtPassword).
- Botón de acceso (btnLogin), que permite iniciar sesión al hacer clic.

- Etiquetas (JLabel) que guían al usuario con los nombres de los campos.
- Paneles (JPanel) con gradientes de color azul y sombras suaves que brindan una apariencia atractiva y elegante.

La vista establece una asociación directa con el controlador LoginControlador, el cual es responsable de procesar las acciones del usuario. Al presionar el botón “INGRESAR”, se activa un evento que el controlador gestiona para validar las credenciales.

También incluye métodos de utilidad como:

- `getUsuario()`: devuelve el texto ingresado en el campo de usuario.
- `getPassword()`: retorna la contraseña ingresada como una cadena de texto.
- `mostrarMensaje(String mensaje)`: muestra cuadros de diálogo informativos o de advertencia para interactuar con el usuario.
- **Clase Main:**

La clase Main pertenece al paquete Vista y representa el punto de entrada principal de la aplicación. Su función principal es inicializar y ejecutar la interfaz gráfica del sistema académico, asegurando que todos los componentes Swing se carguen correctamente dentro del hilo de despacho de eventos (Event Dispatch Thread).

La clase incluye dos opciones de inicio:

- `new LoginVista().setVisible(true);` permite ejecutar la aplicación mostrando primero la ventana de inicio de sesión, ideal cuando se conecta con una base de datos o sistema de autenticación.
- `new VentanaPrincipal().setVisible(true);` ejecuta directamente la ventana principal del sistema, omitiendo el login (útil durante pruebas o desarrollo).

- **Clase VentanaPrincipal:**

La clase VentanaPrincipal pertenece al paquete Vista y representa la interfaz gráfica principal del sistema de gestión académica.

Extiende de JFrame y contiene tres módulos principales:

- **Estudiantes (Ordenamiento):** permite registrar y ordenar estudiantes según diferentes criterios y métodos de ordenamiento.
- **Profesores y Cursos:** gestiona la información de profesores y los cursos asignados.
- **Control de Asistencia:** administra los registros de asistencia de los estudiantes mediante el uso del controlador AsistenciaControlador.

Además, la clase integra la conexión con el controlador, aplicando el patrón Modelo–Vista–Controlador (MVC), y gestiona la interacción entre los datos (modelo) y las acciones del usuario (vista).

Bibliografía:

- Yahya, H., & Anwar, R. M. (2013). *Monitoring student attendance using dashboard*. International Journal of Asian Social Science, 3(9), 2062–2069.
<https://archive.aessweb.com/index.php/5007/article/view/2545>
- Digital Ware. (2023, 10 de septiembre). *Conozca las ventajas de automatizar la gestión de tiempos y turnos en una organización*. Digital Ware.
<https://www.digitalware.com.co/blog/conozca-las-ventajas-de-automatizar-la-gestion-de-tiempos-y-turnos-en-una-organizacion/>
- Gutiérrez Delgado, Walter Rolando (2019), “*Sistema de información para mejorar la gestión académica en el colegio Túpac Amaru de la provincia de Chincheros-Apurímac*”. <https://repositorio.unajma.edu.pe/handle/20.500.14168/552>
- Loyola Cardozo, Trinidad Vanessa Valeska (2019), “*Sistema informático para la gestión de asistencia en la institución educativa integrada Jornada Escolar Completa Santa Teresa, Tarma*”.
https://alicia.concytec.gob.pe/vufind/Record/UPLA_81cf890105ce67ad2f95f3ababdf154c
- Osorio Giraldo, A. M., Osorio Giraldo, M., & Ramírez Cadavid, M. A. (2020). Sistema de gestión académico para instituciones educativas. **Revista CEA**, 6(11), 163-176. <https://revistas.itm.edu.co/index.php/revista-cea/issue/view/101>

- “Eficiencia de la transformación digital en el desempeño administrativo en las escuelas de Piura, Perú” (Alburqueque-Dávila, A., Davis-Carrillo, W., & Esteves-Fajardo, Z., 2024)
<https://www.cienciamatriarevista.org.ve/index.php/cm/article/view/1353>
- Cabero, J., & Llorente, M. C. (2019). *Tecnologías de la información y la comunicación para la enseñanza: nuevas estrategias formativas*. Editorial Síntesis.
- García, L., & Paredes, M. (2021). *La gestión escolar en la era digital: retos y oportunidades para la automatización administrativa*. Revista Iberoamericana de Tecnología Educativa, 15(2), 45–60.
- UNESCO. (2021). *Transformación digital en la educación: hacia sistemas más resilientes e inclusivos*. París: UNESCO Publishing.
- Osorio Giraldo, L., Ramírez, D., & Rojas, E. (2020). *Sistemas integrados de gestión escolar: impacto en la eficiencia administrativa*. Revista Colombiana de Educación y Tecnología, 8(3), 87–101.
- Joyanes Aguilar, M. (2018). *Programación orientada a objetos con Java: un enfoque práctico*. McGraw-Hill Interamericana.
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Estructuras de datos y algoritmos en Java*. Wiley.

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms (4th ed.). MIT Press.
- Weiss, M. A. (2013). Data Structures and Algorithm Analysis in Java (3rd ed.). Pearson.
- Horstmann, C. S. (2019). Core Java Volume I – Fundamentals (11th ed.). Prentice Hall.