UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Sistema de Gestión Educativa para IE.3081 ALMIRANTE MIGUEL GRAU SEMINARIO 2023

AUTORES:

Rios Quijandria Alessandro del Piero (orcid.org/0000-0003-2873-8974)

Llico Labra, Kevin Jesus (orcid.org/0000-0001-5329-0257)

Luna Lopez, Victor Manuel (orcid.org/0000-0003-0252-883X)

Medina Ramos José Guzmaro(orcid.org/0000-0003-1925-7343)

ASESOR:

Mg.JOSUE JOEL RIOS HERRERA

LÍNEA DE INVESTIGACIÓN:

Tecnologías de la Información y comunicación

LIMA NORTE - PERÚ

2023

ÍNDICE

1. Introducción

En la actualidad, la Institución Educativa Pública de nivel Secundario se enfrenta a un desafío crucial: la carencia de un sistema de software que pueda gestionar de manera eficiente a sus alumnos y al personal que labora en sus instalaciones. Esta ausencia de una herramienta tecnológica adecuada ha generado una serie de problemáticas que afectan directamente la operatividad y la calidad de la educación que se ofrece.

En vista de estas problemáticas, la implementación de una aplicación de software diseñada específicamente para las necesidades de la institución educativa se presenta como una solución esencial. Esta herramienta no sólo optimizará la administración interna, sino que también mejorará la experiencia educativa de los estudiantes y fortalecerá la comunicación entre todos los actores involucrados en la comunidad educativa. La inversión en esta tecnología es fundamental para el crecimiento y el éxito continuo de la institución.

2. Desarrollo

2.1. Datos Generales de la Empresa

- Ruc: 20605093877
- Nombre: IE.3081 ALMIRANTE MIGUEL GRAU SEMINARIO.
- Razón social de la empresa: IE.3081 ALMIRANTE MIGUEL GRAU SEMINARIO.
- Teléfono: 568-9668
- Logo:

- Dirección: Jr.Benjamin Quiroga.
- Redes sociales:

- Pagina Web: http://miguelgrau3081.blogspot.pe

2.1.1. Mision y Vision:

Somos una I.E. estatal de Primaria y Secundaria de los ámbitos urbanos de la urb. San Germán del Distrito de San Martín de Porres que forma niños y adolescentes con sólidos valores éticos y morales.Promovemos la adquisición de capacidades de acuerdo al DCN del MED, favoreciendo el aprendizaje de los alumnos para que sean capaces de reconocer y afirmar su identidad; desarrollando habilidades y destrezas que le permita desarrollar su pensamiento crítico, reflexivo y creativo para desenvolverse como ciudadanos responsables dentro de su comunidad, asumiendo su compromiso e identificación con el desarrollo institucional, comunal, regional y nacional.

Al año 2015 aspiramos ser una I.E. que brinde una educación de calidad, con docentes calificados que forman niños y adolescentes integrales, desarrollando sus capacidades y actitudes que permitan ser personas con autoestima elevada, proactivos y competentes; convirtiéndose en promotores del desarrollo de la comunidad dentro de la práctica de valores para una cultura de paz, capaces de enfrentar los retos de la globalización y los avances científicos-tecnológicos.

2.1.2. Objetivos Generales y Específicos

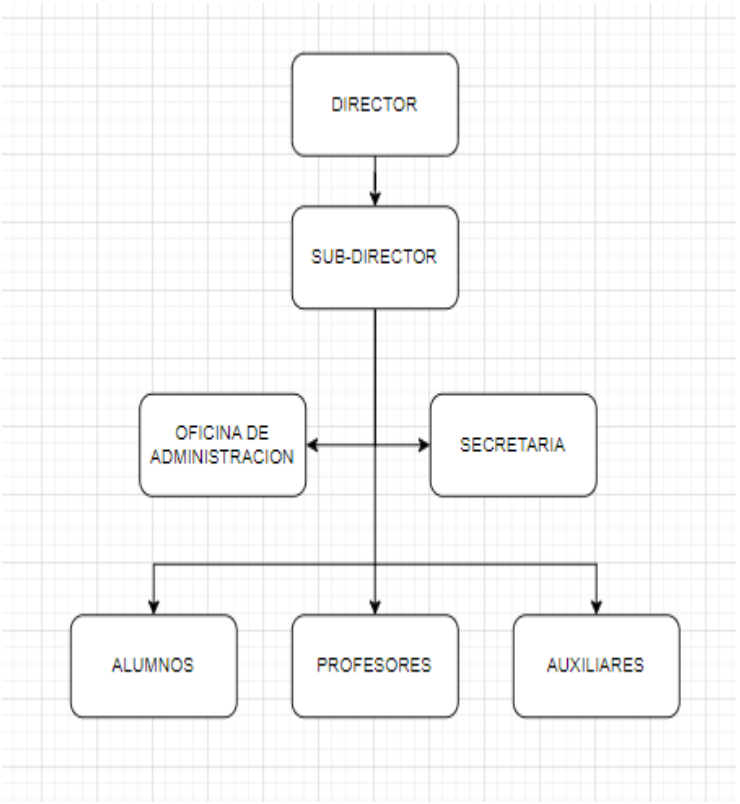
GENERAL:

- Brindar educación de alta calidad.

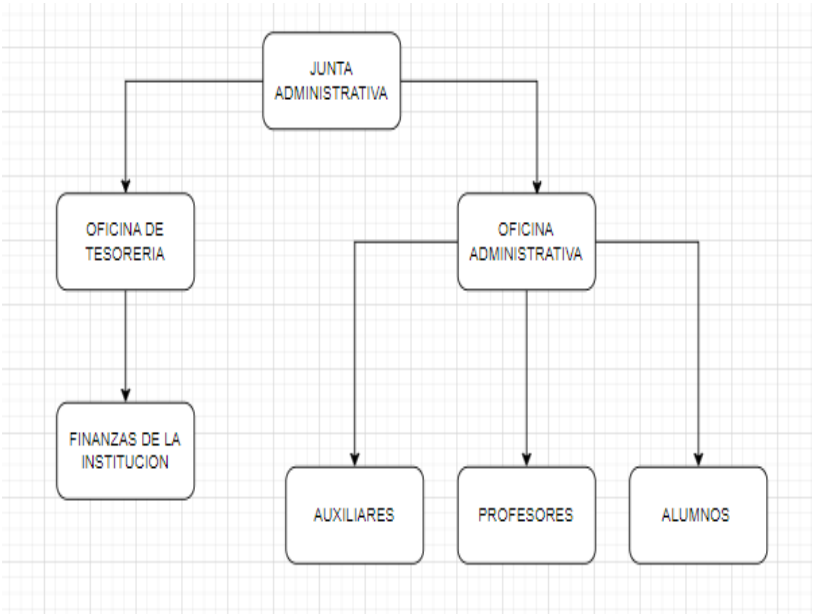
ESPECÍFICOS:

- Implementar servicios educativos sociales.
- Generar conocimientos y competencias educativas de escolares en el distrito de San Martín de Porres.

2.1.3. Organigrama:



2.1.4. Funciones de Mérito Organigrama:



2.2. Realidad Problemática:

La Institución Almirante Miguel Grau se ve afectada por diversos desafíos, ya que no dispone de un sistema completo de gestión. La organización académica está desordenada, la comunicación es poco efectiva, la inscripción de estudiantes y el seguimiento de asistencias se realizan de forma manual, la planificación de

currículos es complicada, existe preocupación por la seguridad de los datos y la administración enfrenta problemas de eficiencia.

2.3. Selección del Problema:

Para la selección de la problemática se tomó en cuenta los factores que la institución necesitaba fortalecer lo antes posible, como lo es la administración de sus alumnos y todos los procesos que esto conlleva, como el cobro de matrículas, pensiones,gestión de notas de cada alumno, etc. La falta de este sistema hace que la institución pierda u omita información relevante para su funcionamiento.

3. Propuesta de Solución

3.1. Objetivo general

- Implementar un Sistema de Gestión Educativa efectivo en la Institución Educativa N° 3081 "Almirante Miguel Grau Seminario" durante el año 2023, con el propósito de optimizar la administración y el desarrollo educativo en todos sus niveles.

3.2. Objetivo Específico

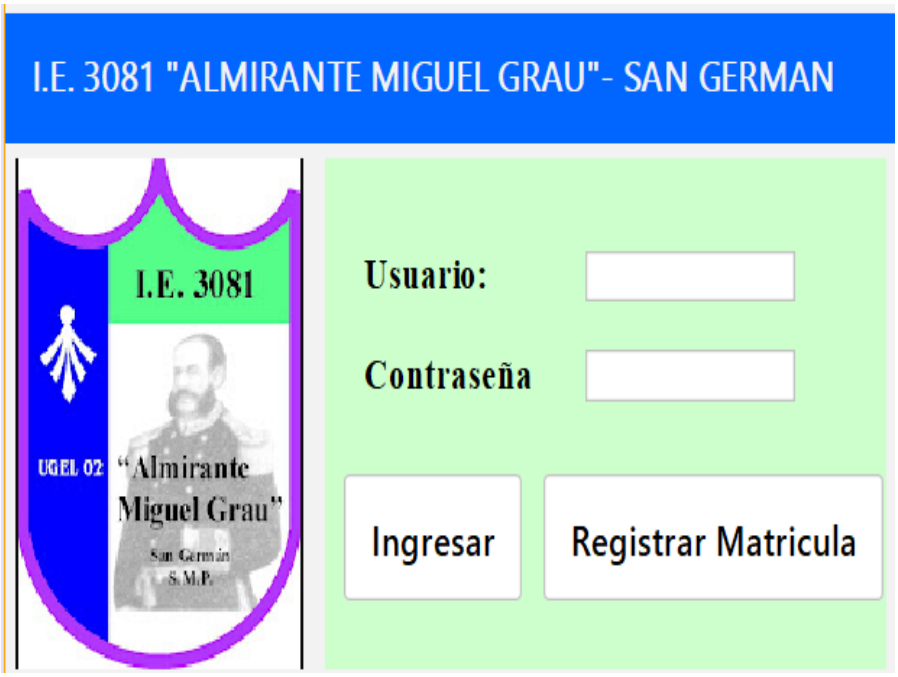
- Desarrollar e implementar módulos específicos que permitan automatizar procesos de la institución Almirante Miguel Grau, como la matrícula, el control de asistencias, la gestión de recursos, y la generación de informes, con el fin de reducir la carga de trabajo manual y mejorar la eficiencia en la gestión de recursos.

- Establecer un sistema que involucre a padres, docentes y estudiantes, facilitando el intercambio de información relevante sobre el progreso académico, matrículas, información del personal del centro educativo, promoviendo así una participación activa y comprometida de la comunidad educativa.

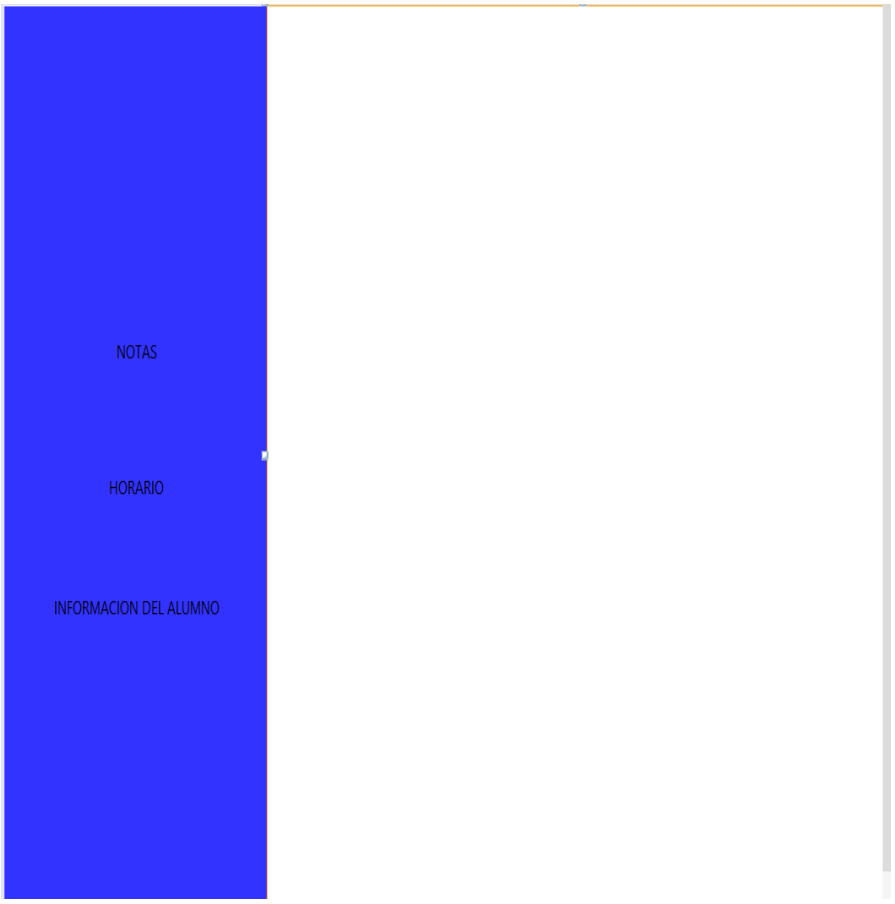
3.3. Propuesta

3.3.1. Visual

Ventana Login:



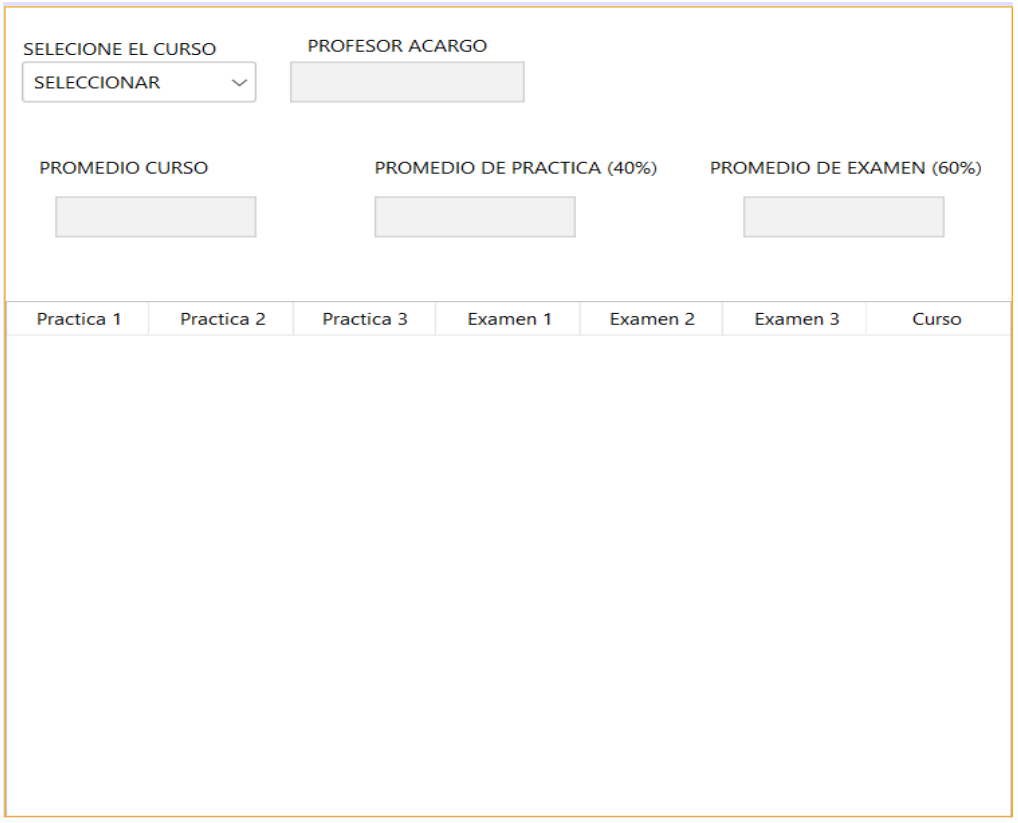
Ventana Principal del Alumno:



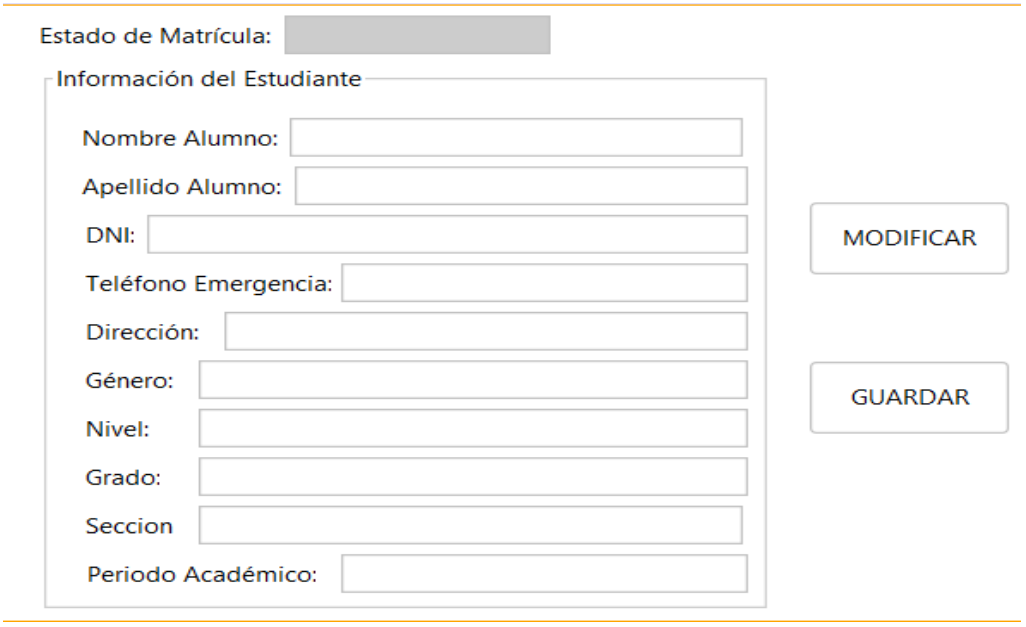
Ventana Registro de Matrícula

Ventana Horario Alumno:

Ventana Notas del Alumno:



Ventana Informacion del Alumno:



FRM.aulas

Actividad.java

```
public class Actividad {
    private int id;
    private Curso curso;
    private String descripcion;

    public Actividad() {
    }


    public Actividad(int id, Curso curso, String descripcion) {
        this.id = id;
        this.curso = curso;
        this.descripcion = descripcion;
    }


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Curso getCurso() {
        return curso;
    }
    public void setCurso(Curso curso) {
        this.curso = curso;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }
```

Alumno.java

```
public class Alumno extends Persona {

    private int idAlumno;
    private int grado;
    private String nivel;
    private int idUsuario;

    public Alumno() {
    }

    public Alumno(int idAlumno, int grado, String nivel, int idUsuario) {
        this.idAlumno = idAlumno;
        this.grado = grado;
        this.nivel = nivel;
        this.idUsuario = idUsuario;
    }

    public Alumno(int idAlumno, int grado, String nivel,int idUsuario, String nombre, String apellido, String direccion, 
        super(nombre, apellido, direccion, documentoIdentidad, telefono, genero);
        this.idAlumno = idAlumno;
        this.grado = grado;
        this.nivel = nivel;
        this.idUsuario = idUsuario;
    }

    public int getIdAlumno() {
        return idAlumno;
    }

    public void setIdAlumno(int idAlumno) {
        this.idAlumno = idAlumno;
    }

    public int getGrado() {
        return grado;
    }

    public String getNivel() {
        return nivel;
    }

    public void setNivel(String nivel) {
        this.nivel = nivel;
    }

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    @Override
    public String toString() {
        return "Alumno{" + "idAlumno=" + idAlumno + ", grado=" + grado + ", nivel=" + nivel + ", idUsuario=" + idUsuario +
    }



}
```

Aula.java

```
public class Aula {
    private int id;
    private String numeroAula;
    private String grado;
    private String nivel;
    private String seccion;
    private int capacidadAlumnos;

    public Aula() {
    }

    public Aula(int id, String numeroAula, String grado, String nivel, String seccion, int capacidadAlumnos) {
        this.id = id;
        this.numeroAula = numeroAula;
        this.grado = grado;
        this.nivel = nivel;
        this.seccion = seccion;
        this.capacidadAlumnos = capacidadAlumnos;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNumeroAula() {
        return numeroAula;
    }

    public void setNumeroAula(String numeroAula) {
        this.numeroAula = numeroAula;
    }
```

```
    public String getGrado() {
        return grado;
    }

    public void setGrado(String grado) {
        this.grado = grado;
    }

    public String getNivel() {
        return nivel;
    }

    public void setNivel(String nivel) {
        this.nivel = nivel;
    }

    public String getSeccion() {
        return seccion;
    }

    public void setSeccion(String seccion) {
        this.seccion = seccion;
    }

    public int getCapacidadAlumnos() {
        return capacidadAlumnos;
    }

    public void setCapacidadAlumnos(int capacidadAlumnos) {
        this.capacidadAlumnos = capacidadAlumnos;
    }

    @Override
    public String toString() {
        return "Aula{" + "id=" + id + ", numeroAula=" + numeroAula + ", grado=" + grado + ", nivel=" + nivel + ", seccion
    }
```

Curso.java

```
public class Curso {
    private int idCurso;
    private String nombreCurso;
    private int idProfesor;
    private String nombreProfesor;

    public Curso() {
    }

    public Curso(int id, String nombreCurso) {
        this.idCurso = id;
        this.nombreCurso = nombreCurso;
    }

    public int getIdCurso() {
        return idCurso;
    }

    public void setIdCurso(int idCurso) {
        this.idCurso = idCurso;
    }

    public String getNombreCurso() {
        return nombreCurso;
    }

    public void setNombreCurso(String nombreCurso) {
        this.nombreCurso = nombreCurso;
    }

    public int getIdProfesor() {
        return idProfesor;
    }

    public void setIdProfesor(int idProfesor) {
        this.idProfesor = idProfesor;
```

```
    public String getNombreProfesor() {
        return nombreProfesor;
    }

    public void setNombreProfesor(String nombreProfesor) {
        this.nombreProfesor = nombreProfesor;
    }


    @Override
    public String toString() {
        return "Curso{" + "idCurso=" + idCurso + ", nombreCurso=" + nombreCurso + ", idProfesor=" + idProfesor + ", nombre
    }


}
```

Matricula.java

```
public class Matricula {
    private int id;
    private int idAlumno;
    private Date fechaMatricula;
    private double costo;
    private String metodoPago;
    private String periodoAcademico;
    private String estadoMatricula;
    private double descuento;

    public Matricula() {
    }

    public Matricula(int id, int idAlumno, Date fechaMatricula, double costo, String metodoPago, String periodoAcademico,
        this.id = id;
        this.idAlumno = idAlumno;
        this.fechaMatricula = fechaMatricula;
        this.costo = costo;
        this.metodoPago = metodoPago;
        this.periodoAcademico = periodoAcademico;
        this.estadoMatricula = estadoMatricula;
        this.descuento = descuento;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getIdAlumno() {
        return idAlumno;
```

```
    public void setIdAlumno(int idAlumno) {
        this.idAlumno = idAlumno;
    }

    public Date getFechaMatricula() {
        return fechaMatricula;
    }

    public void setFechaMatricula(Date fechaMatricula) {
        this.fechaMatricula = fechaMatricula;
    }

    public double getCosto() {
        return costo;
    }

    public void setCosto(double costo) {
        this.costo = costo;
    }

    public String getMetodoPago() {
        return metodoPago;
    }

    public void setMetodoPago(String metodoPago) {
        this.metodoPago = metodoPago;
    }

    public String getPeriodoAcademico() {
        return periodoAcademico;
    }

    public void setPeriodoAcademico(String periodoAcademico) {
        this.periodoAcademico = periodoAcademico;
    }
```

```
    public String getEstadoMatricula() {
        return estadoMatricula;
    }

    public void setEstadoMatricula(String estadoMatricula) {
        this.estadoMatricula = estadoMatricula;
    }

    public double getDescuento() {
        return descuento;
    }

    public void setDescuento(double descuento) {
        this.descuento = descuento;
    }



    @Override
    public String toString() {
        return super.toString(); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/OverriddenMethod
    }
}
```

Nota.java

```
public class Nota {

    private int id;
    private double pc1;
    private double pc2;
    private double pc3;
    private double exa1;
    private double exa2;
    private double exa3;

    public Nota() {
    }

    public Nota(int id, double pc1, double pc2, double pc3, double exa1, double exa2, double exa3) {
        this.id = id;
        this.pc1 = pc1;
        this.pc2 = pc2;
        this.pc3 = pc3;
        this.exa1 = exa1;
        this.exa2 = exa2;
        this.exa3 = exa3;
    }

    public double calcularPromedioPracticas(){
        return (pc1+pc2+pc3)/3;
    }

    public double calcularPromedioExamenes(){
        return (pc1+pc2+pc3)/3;
    }

    public int getId() {
        return id;
    }
```

```
public void setId(int id) {
    this.id = id;
}

public double getPc1() {
    return pc1;
}

public void setPc1(double pc1) {
    this.pc1 = pc1;
}

public double getPc2() {
    return pc2;
}

public void setPc2(double pc2) {
    this.pc2 = pc2;
}

public double getPc3() {
    return pc3;
}

public void setPc3(double pc3) {
    this.pc3 = pc3;
}

public double getExa1() {
    return exa1;
}

public void setExa1(double exa1) {
    this.exa1 = exa1;
}
```

```
public double getExa2() {
    return exa2;
}

public void setExa2(double exa2) {
    this.exa2 = exa2;
}

public double getExa3() {
    return exa3;
}

public void setExa3(double exa3) {
    this.exa3 = exa3;
}
```

persona.java

```
public abstract class Persona {

    private String nombre;
    private String apellido;
    private String direccion;
    private String documentoIdentidad;
    private String telefono;
    private char genero;

    public Persona() {
    }

    public Persona(String nombre, String apellido, String direccion, String documentoIdentidad, String telefono, char gene
        this.nombre = nombre;
        this.apellido = apellido;
        this.direccion = direccion;
        this.documentoIdentidad = documentoIdentidad;
        this.telefono = telefono;
        this.genero = genero;
    }

    public char getGenero() {
        return genero;
    }

    public void setGenero(char genero) {
        this.genero = genero;
    }


    public String getNombre() {
        return nombre;
    }
```

```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public String getDocumentoIdentidad() {
        return documentoIdentidad;
    }

    public void setDocumentoIdentidad(String documentoIdentidad) {
        this.documentoIdentidad = documentoIdentidad;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }
```

```
    @Override
    public String toString() {
        return "Persona{" + "nombre=" + nombre + ", apellido=" + apellido + ", direccion=" + direccion + ", documentoIdent
    }

}
```

profesor.java

```
public class Profesor extends Persona{
    private int idProfesor;
    private int idUsuario;

    public Profesor() {
    }

    public int getIdProfesor() {
        return idProfesor;
    }

    public void setIdProfesor(int idProfesor) {
        this.idProfesor = idProfesor;
    }

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    @Override
    public String toString() {
        return "Profesor{" + "idProfesor=" + idProfesor + ", idUsuario=" + idUsuario + '}';
    }
```

usuario.java

```
public class Usuario {
    private int id;
    private String nombreUsuario;
    private String password;
    private int tipoUsuario;

    public Usuario() {
    }

    public Usuario(int id, String nombreUsuario, String password, int tipoUsuario) {
        this.id = id;
        this.nombreUsuario = nombreUsuario;
        this.password = password;
        this.tipoUsuario = tipoUsuario;
    }

    public int getTipoUsuario() {
        return tipoUsuario;
    }

    public void setTipoUsuario(int tipoUsuario) {
        this.tipoUsuario = tipoUsuario;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getNombreUsuario() {
        return nombreUsuario;
    }

    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;
    }

    public String getPassword() {
        return password;
    }
```

3.3.2. Código

A continuación se procederá a explicar las estructuras de datos que se van a utilizar en el presente trabajo:

Listas Enlazadas Simples: Una lista enlazada es una colección lineal donde se presentan elementos llamados nodos. Hay un orden entre ellos que se establece con cursor; dirección o referencias a otros nodos.

```
public class ListaEnlazadaSimple<T> implements IListaEnlazadaSimple<Object> {

    private NodoSimple inicio;
    private NodoSimple fin;

    public ListaEnlazadaSimple() {
        inicio = fin = null;
    }

    //agregar al final
    @Override
    public void agregar(Object obj) {
        NodoSimple nuevo = new NodoSimple(o: obj);
        if (estaVacia()) {
            inicio = nuevo;
            fin = nuevo;
        } else {
            fin.setSiguiente(siguiente:nuevo);
            fin = nuevo;
        }
    }

    public void agregarAlInicio(Object obj) {
        NodoSimple nuevo = new NodoSimple(o: obj);
        nuevo.setSiguiente(siguiente:this.inicio);
        inicio = nuevo;
    }
```

```
public void eliminar(Object obj) {
    NodoSimple aux = inicio;
    NodoSimple eliminado = new NodoSimple();
    eliminado.setDato(dato: obj);

    if (eliminado.getDato().equals(obj:inicio.getDato())) {
        eliminarInicio();
        System.out.println(x: "Se elimino al principio de la lista");
    } else {
        while (aux != null && aux.getSiguiente() != null) {
            if (aux.getSiguiente().getDato().equals(obj:eliminado.getDato())) {
                // Elimina el nodo siguiente al nodo actual
                aux.setSiguiente(siguiente:aux.getSiguiente().getSiguiente());
            }
            aux = aux.getSiguiente();
        }
    }
}

//eliminar al inicio
public void eliminarInicio() {

    if (estaVacia()) {
        System.out.println(x: "Lista Vacia");
    } else {
        NodoSimple aux;
        aux = inicio.getSiguiente();
        inicio.setSiguiente(siguiente:null);
        inicio = aux;
    }
}
```

```
74  public void mostrar() {
75      if (estaVacia()) {
76          System.out.println(x: "lista vacia");
77      } else {
78          NodoSimple aux = inicio;
79          while (aux != null) {
80              System.out.println(x: aux.toString());
81              aux = aux.getSiguiente();
82          }
83      }
84
85      @Override
87      public void actualizar(Object objViejo, Object objNuevo) {
88          NodoSimple aux = (NodoSimple) buscar(obj:objViejo);
89
90          if (aux != null) {
91              aux.setDato(dato: objNuevo);
92          } else {
93              System.out.println(x: "El dato no existe");
94          }
      }
```

```java
public Object buscar(Object obj) {
    NodoSimple aux = inicio;
    //Va tomar el nodo el cual tenga dentro el dato correspondiente
    NodoSimple encontrado = null;

    while (aux != null) {
        if (aux.getDato().equals(obj)) {
            encontrado = aux;
        }
        aux = aux.getSiguiente();
    }
    return encontrado;
}

public String mensajeByBuscar(Object obj){
    NodoSimple encontrado = (NodoSimple) buscar(obj);
    String mensaje;

    if(encontrado!=null){
        mensaje="Se elimino:"+encontrado.getDato().toString();
    }else
        mensaje="No existe en la lista";

    return mensaje;
}
```

Listas Enlazadas Dobles: Una lista doblemente enlazada es una lista lineal en la que cada nodo tiene dos enlaces, uno al siguiente nodo y otro al nodo anterior.

```java
public class ListaEnlazadaDoble<Object> implements IListaEnlazadaDoble<Object> {

    private NodoDoble inicio;
    private NodoDoble fin;

    public ListaEnlazadaDoble() {
        inicio = fin = null;
    }

    //agregar al final
    @Override
    public void agregar(Object obj) {
        NodoDoble nuevo = new NodoDoble();
        nuevo.setDato(dato: obj);

        if (estaVacia()) {
            inicio = nuevo;
            fin = nuevo;
        } else {
            nuevo.setAnterior(anterior: fin);
            fin.setSiguiente(siguiente:nuevo);
            fin = nuevo;
        }
    }
```

```
    public void eliminar(Object obj) {
40      NodoDoble aux = new NodoDoble(dato: obj);
41      NodoDoble eliminado = (NodoDoble) buscar((Object) aux);
42
43      if (!estaVacia() && eliminado != null) {
44          //Si el objeto a eliminar es el primero
45          if (eliminado.getDato().equals(obj:inicio.getDato())) {
46              eliminarInicio();
47              return;
48          }
49          //Si el dato a eliminar es el primero
50          if (eliminado.getDato().equals(obj:fin.getDato())) {
51              eliminarFinal();
52              return;
53          }
54
55          /*
56          Tomamos el nodo anterior del nodo que queremos eliminar y le seteamos
57          el siguiente nodo al cual apunta el nodo que queremos eliminar
58           */
59          eliminado.getAnterior().setSiguiente(siguiente:eliminado.getSiguiente());
60          /*
61          hacemos que su siguiente nodo apunte a null y asi quede liberado
62           */
63          eliminado.setSiguiente(siguiente:null);
64          /*
65          hacemos que su anterior nodo apunte a null y asi quede liberado
66           */
67          eliminado.setAnterior(anterior: null);
68          /*
69          Se veria asi
70          null<-NODO_ELIMNADO->null
71           */
72      }
73  }
```

```
75      public void eliminarInicio() {
76          NodoDoble aux = inicio.getSiguiente();
77          inicio.setSiguiente(siguiente:null);
78          inicio = aux;
79          inicio.setAnterior(anterior: null);
80      }
81
82      public void eliminarFinal() {
83          NodoDoble aux = fin.getAnterior();
84          fin.setAnterior(anterior: null);
85          fin = aux;
86          fin.setSiguiente(siguiente:null);
87      }
88
89      //mostrar de principio a fin
90      @Override
91      public void mostrar() {
92          NodoDoble aux = inicio;
93          if (estaVacia()) {
94              System.out.println(x: "Lista Vacia");
95          }
96          while (aux != null) {
97              System.out.println(x: aux.toString());
98              aux = aux.getSiguiente();
99          }
100     }
101
102     @Override
103     public void actualizar(Object objAntiguo, Object objNuevo) {
104         NodoDoble aux = (NodoDoble) buscar(obj:objAntiguo);
105
106         if (aux != null) {
107             aux.setDato(dato: objNuevo);
108         } else {
109             System.out.println(x: "El dato no existe");
110         }
111     }
```

```
113     @Override
114     public Object buscar(Object obj) {
115         NodoDoble aux = inicio;
116         NodoDoble encontrado = null;
117 
118         while (aux != null) {
119             if (aux.getDato().equals(obj)) {
120                 encontrado = aux;
121             }
122             aux = aux.getSiguiente();
123         }
124 
125         return (Object) encontrado;
126     }
127 
128     public String mensajeByBuscar(Object obj) {
129         NodoDoble encontrado = (NodoDoble) buscar(obj);
130         String mensaje;
131 
132         if (encontrado != null) {
133             mensaje = "Se encontro:" + encontrado.toString();
134         } else {
135             mensaje = "No existe en la lista";
136         }
137 
138         return mensaje;
139     }
140 
141     @Override
142     public boolean estaVacia() {
143         return inicio == null;
144     }
```

Listas Circulares: En una lista enlazada circular, el primer nodo y el último nodo están enlazados entre sí. Para recorrer una lista enlazada circular, podemos comenzar desde cualquier nodo,en cualquier dirección hasta regresar al nodo original.

```java
public class ListaEnlazadaCircularSimple<T> implements IListaEnlazadaSimple<Object> {

    private NodoSimple inicio, fin;


    public ListaEnlazadaCircularSimple() {
        inicio = fin = null;
    }

    @Override
    public void agregar(Object obj) {
        NodoSimple nuevo = new NodoSimple(o: obj);
        if (estaVacia()) {
            inicio = fin = nuevo;
            inicio.setSiguiente(siguiente:nuevo);
        } else {
            fin.setSiguiente(siguiente:nuevo);
            fin = nuevo;
            fin.setSiguiente(siguiente:inicio);
        }
    }
```

```java
    public void eliminar(Object obj) {
        if (estaVacia()) {
            System.out.println(x: "Lista Vacia");
            return;
        }

        if (inicio.getDato().equals(obj)) {
            eliminarInicio();
            return;
        }

        if (fin.getDato().equals(obj)) {
            eliminarFinal();
            return;
        }

        NodoSimple aux = inicio;
        do {
            if (aux.getSiguiente().getDato().equals(obj)) {
                aux.setSiguiente(siguiente:aux.getSiguiente().getSiguiente());
                return;
            }
            aux = aux.getSiguiente();
        } while (aux != inicio);

    }
```

```java
    public void eliminarInicio() {
        if (estaVacia()) {
            System.out.println(x: "Lista Vacia");
        } else {
            NodoSimple aux = inicio.getSiguiente();
            inicio.setSiguiente(siguiente:null);
            inicio = aux;
            fin.setSiguiente(siguiente:inicio);
        }
    }

    public void eliminarFinal() {
        if (estaVacia()) {
            System.out.println(x: "Lista Vacia");
        } else {
            NodoSimple aux = inicio;
            do {
                if (aux.getSiguiente().equals(obj:fin)) {
                    fin.setSiguiente(siguiente:null);
                    fin = aux;
                    fin.setSiguiente(siguiente:inicio);
                }
                aux = aux.getSiguiente();
            } while (aux != inicio);
        }
    }
```

```java
    public void mostrar() {
        NodoSimple aux = inicio;

        do {
            System.out.println(x: aux.toString());
            aux = aux.getSiguiente();
        } while (aux != inicio && aux != null);
    }

    @Override
    public void actualizar(Object objAntiguo, Object objNuevo) {
        NodoSimple aux = inicio;
        do {
            if (aux.getDato().equals(obj:objAntiguo)) {
                aux.setDato(dato: objNuevo);
            }
            aux = aux.getSiguiente();
        } while (aux != inicio);
    }

    //Busca el dato dentro de los nodos y devuelve el nodo
    @Override
    public Object buscar(Object obj) {
        NodoSimple aux = inicio;
        NodoSimple encontrado = null;
        while (aux != null) {
            if (aux.getDato().equals(obj)) {
                encontrado = aux;
            }
            aux = aux.getSiguiente();
        }
        return encontrado;
    }

    @Override
    public boolean estaVacia() {
        return inicio == null;
```

Pilas: Es una estructura de datos lineal basada en el principio LIFO (último en entrar, primero en salir), es decir, último en entrar, primero en salir.

Una pila se comporta como una caja que contiene múltiples elementos, donde sólo se pueden realizar dos operaciones: empujar y empujar. Apilar un elemento lo agrega a la parte superior de la pila y desapilar lo elimina de la parte superior de la pila.

```
10  public class Pila implements IPila<Object>{
11
12      private NodoSimple cima;
13
14      public Pila() {
15          cima=null;
16      }
17
18      @Override
19      public void push(Object dato) {
20          NodoSimple nuevo = new NodoSimple(o: dato);
21          nuevo.setSiguiente(siguiente:cima);
22          cima=nuevo;
23      }
24
25      @Override
26      public void pop() {
27          if(estaVacia())
28              System.out.println(x: "Pila Vacia");
29          else{
30              cima=cima.getSiguiente();
31          }
32      }
33
34      @Override
35      public boolean estaVacia() {
36          return cima==null;
37      }
38
39      @Override
40      public void limpiar() {
41          while(!estaVacia()){
42              pop();
43          }
44      }
45
```

Colas: En programación, una cola es una estructura de datos lineal basada en el principio FIFO (primero en entrar, primero en salir). Una fila actúa como un cuadro con múltiples elementos donde sólo se pueden realizar dos operaciones: fila y fila. Una cola agrega elementos al final de la cola, mientras que al eliminar una cola se eliminan elementos del frente de la cola.

```
10  public class Cola implements ICola<Object> {
11
12      private NodoSimple inicio,fin;
13
14      public Cola() {
15          inicio=fin=null;
16      }
17
18      @Override
19      public void push(Object dato) {
20          NodoSimple nuevo = new NodoSimple(o: dato);
21          if(estaVacia())
22              inicio=fin=nuevo;
23          else{
24              fin.setSiguiente(siguiente:nuevo);
25              fin=nuevo;
26          }
27      }
28
29      @Override
30      public void pop() {
31          if(estaVacia())
32              System.out.println(x: "Lista vacia");
33          else{
34              inicio=inicio.getSiguiente();
35          }
36      }
37
38      @Override
39      public boolean estaVacia() {
40          return inicio==null;
41      }
```

```
@Override
public void limpiar() {
    while(!estaVacia()){
        pop();
        fin=null;
    }
}
```

Grafos No Dirigidos: Un grafo es un conjunto de puntos y líneas que unen pares de esos puntos, En los grafos no dirigidos las aristas son doble mano (se puede ir en ambos sentidos).

```java
package javaapplication12;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class Graph {
    private int vertices;
    private LinkedList<Integer> adj[];
    private String pathResult;

    public Graph(int vertices) {
        this.vertices = vertices;
        adj = new LinkedList[vertices];

        for (int i = 0; i < vertices; ++i)
            adj[i] = new LinkedList();
    }

    public void addEdge(int v, int w) {
        adj[v].add(e: w);
        adj[w].add(e: v);  // Agregar la conexión en ambas direcciones para hacerla no dirigida
    }

    public void findPath(int start, int end) {
        List<Integer> path = new ArrayList<>();
        boolean[] visited = new boolean[vertices];

        pathResult = dfs(current: start, end, visited, path);

        if (pathResult.isEmpty()) {
            pathResult = "Debes tomar las escaleras. No hay camino disponible.";
        }
    }
```

```java
    public void findPath(int start, int end) {
        List<Integer> path = new ArrayList<>();
        boolean[] visited = new boolean[vertices];

        pathResult = dfs(current: start, end, visited, path);

        if (pathResult.isEmpty()) {
            pathResult = "Debes tomar las escaleras. No hay camino disponible.";
        }
    }
    private void connectGroup(int start, int end) {
        for (int i = start; i <= end; i++) {
            for (int j = i + 1; j <= end; j++) {
                addEdge(i - 1, j - 1);
            }
        }
    }

    private void connectWithinGroup(int start, int end) {
        // Lógica para conectar aulas consecutivas dentro de cada grupo
        for (int i = start; i < end; i++) {
            addEdge(i - 1, w: i);
        }
    }
    public void configureConnections() {
        connectGroup(start: 1, end:10);
        connectGroup(start: 11, end:21);
        connectGroup(start: 22, end:33);

        connectWithinGroup(start: 1, end:10);
        connectWithinGroup(start: 11, end:21);
        connectWithinGroup(start: 22, end:33);
    }
```

```
    private String dfs(int current, int end, boolean[] visited, List<Integer> path) {
        visited[current] = true;
        path.add(e: current);

        if (current == end) {
            return buildPathString(path);
        }

        for (int neighbor : adj[current]) {
            if (!visited[neighbor]) {
                String remainingPath = dfs(current: neighbor, end, visited, path);
                if (!remainingPath.isEmpty()) {
                    return remainingPath;
                }
            }
        }

        // Si no encontramos el camino desde el nodo actual, retrocedemos
        path.remove(path.size() - 1);
        return "";
    }

    private String buildPathString(List<Integer> path) {
        StringBuilder pathString = new StringBuilder(str:"Aulas en el camino: ");
        for (int aula : path) {
            pathString.append(aula + 1).append(str:" ");
        }
        return pathString.toString();
    }

    public String getPathResult() {
        return pathResult;
    }

}
```

informaciónAlumno.java

```
public class InformacionAlumno {
    private Matricula matricula;
    private Alumno alumno;
    private Aula aula;

    public InformacionAlumno() {
    }


    public Matricula getMatricula() {
        return matricula;
    }

    public void setMatricula(Matricula matricula) {
        this.matricula = matricula;
    }

    public Alumno getAlumno() {
        return alumno;
    }

    public void setAlumno(Alumno alumno) {
        this.alumno = alumno;
    }

    public Aula getAula() {
        return aula;
    }

    public void setAula(Aula aula) {
        this.aula = aula;
    }
```

notasAlumno.java

```
public class Notas_Cursos_Alumno {
    private Nota nota;
    private Curso curso;

    public Notas_Cursos_Alumno() {
    }

    public Notas_Cursos_Alumno(Nota nota, Curso curso) {
        this.nota = nota;
        this.curso = curso;
    }

    public Nota getNota() {
        return nota;
    }

    public void setNota(Nota nota) {
        this.nota = nota;
    }

    public Curso getCurso() {
        return curso;
    }

    public void setCurso(Curso curso) {
        this.curso = curso;
    }
```

conexiónDB.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
/**
 *
 * @a uthor aless
 */
public class ConexionDB {
    Connection conn;

    public Connection conectarDB() {
        String dbUrl = "jdbc:sqlserver://localhost:1433;databaseName=DB_SisGesAcademica;user=piero;password=root;encrypt=f

        try {
            //Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            conn = DriverManager.getConnection(url:dbUrl);
            System.out.println(x: "conexion exitosa");
        } catch (SQLException e) {
            System.out.println(x: e.toString());
        }
//        catch (ClassNotFoundException ex) {
//            Logger.getLogger(ConexionDB.class.getName()).log(Level.SEVERE, null, ex);
//        }
        return conn;
    }
```

pilas.java

```
public class Pila implements IPila<Object>{

    private NodoSimple cima;

    public Pila() {
        cima=null;
    }

    @Override
    public void push(Object dato) {
        NodoSimple nuevo = new NodoSimple(dato);
        nuevo.setSiguiente(cima);
        cima=nuevo;
    }

    @Override
    public void pop() {
        if(estaVacia())
            System.out.println(x: "Pila Vacia");
        else{
            cima=cima.getSiguiente();
        }
    }

    @Override
    public boolean estaVacia() {
        return cima==null;
    }

    @Override
    public void limpiar() {
        while(!estaVacia()){
            pop();
        }
    }

    public NodoSimple getCima() {
        return cima;
    }

    public void setCima(NodoSimple cima) {
        this.cima = cima;
    }

    @Override
    public String toString() {
        return "Pila{" + "cima=" + cima + '}';
    }
```

4. Conclusiones

- Se concluye que la implementación de un sistema de gestión educativa optimiza la administración y el desarrollo educativo en todos sus niveles, contribuyendo así a una mejora integral en la eficiencia y eficacia de los procesos educativos.

- Se concluye que implementar módulos específicos automatizará los procesos de la institución Almirante Miguel Grau, como la matrícula, el control de asistencias, la gestión de recursos, y la generación de informes, con el fin de reducir la carga de trabajo manual y mejorar la eficiencia en la gestión de recursos.

- Se concluye que establecer un sistema que involucre a padres, docentes y estudiantes, facilita el intercambio de información relevante sobre el progreso académico, matrículas, información del personal del centro educativo, y promueve una participación activa y comprometida de la comunidad educativa.

5. Referencias

- MENESES, Erika. *ESTRUCTURAS DE DATOS* [en línea]. 21 de septiembre de 29 [consultado el 9 de diciembre de 2023]. Ciudad de México. Disponible en: https://www.uv.mx/personal/ermeneses/files/2021/08/Clase5-ListasEnlazadasFinal.pdf

- POZO, Salvador. Capítulo 5 Listas doblemente enlazadas. *Con Clase | Cursos y recursos* [en línea]. [sin fecha] [consultado el 10 de diciembre de 2023]. Disponible en: https://conclase.net/c/edd/cap5

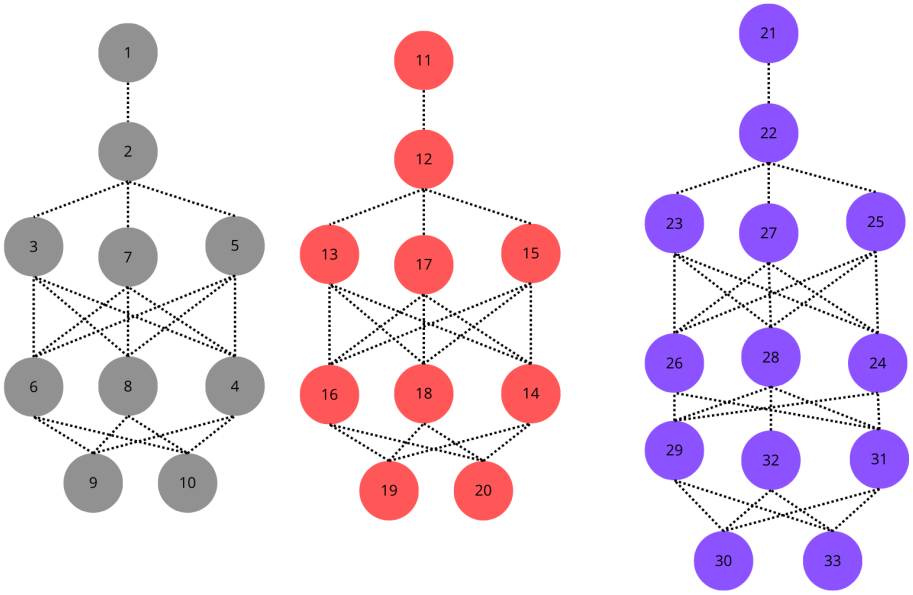
- GUTIERREZ, Gustavo. *Estructura de Datos Unidad 2: Listas* [en línea]. 10 de marzo de 23 [consultado el 9 de diciembre de 2023]. Ciudad de México. Disponible en: https://www.fcca.umich.mx/descargas/apuntes/academia%20de%20informatica/Estructura%20de%20Datos%20%20%20%20G.a.G.C/Unidad%202.pdf

- CASTRO, Steven. ¿Qué es una cola en programación? *Fundamentos de la Programación* [en línea]. 13 de enero de 2023 [consultado el 10 de diciembre de 2023]. Disponible en: https://www.fundamentosprogramacion.site/2023/01/que-es-una-cola-en-programacion.html

- SCALAR, Melanie. *Grafos: una no tan breve introducción* [en línea]. 17 de noviembre de 7 [consultado el 9 de diciembre de 2023]. Buenos Aires. Disponible en: https://www.oia.unsam.edu.ar/wp-content/uploads/2013/10/Grafos.pdf

- I.E. N° 3081 ALMIRANTE MIGUEL GRAU SEMINARIO. *I.E. N° 3081 ALMIRANTE MIGUEL GRAU SEMINARIO* [en línea]. [sin fecha] [consultado el 11 de diciembre de 2023]. Disponible en: https://miguelgrau3081.blogspot.com/

6. Anexos


1
2
3
7
5
6
8
4
9
10
11
12
13
17
15
16
18
14
19
20
21
22
23
27
25
26
28
24
29
32
31
30
33


UGEL-02
I.E. N° 3081
ALMIRANTE MIGUEL GRAU SEMINARIO
"Ingresamos Para Estudiar, Salimos Para Triunfar"