Universidad Mayor de San Andrés
Facultad de Ciencias Puras y Naturales
Carrera de Informática

Sistema para Gestión de Becas

Integrantes :

Univ. Camacho Estivariz Adriana Raquel	C.I. 9869524 LP
Univ. Cruz Mamani Yojana Ariola	C.I. 13342212 LP
Univ. Fernandez Uzquiano Manuel Alejandro	C.I. 12420071 LP
Univ. Iriarte Huanca Rodriguez Josue Ricardo	C.I. 9101092 LP
Univ. Mamani Mamani Grover Marcelo	C.I. 9241044 LP
Univ. Nieva Montalvo Pablo Humberto	C.I. 8306320 LP
Univ. Tito Bravo Joel Fidel	C.I. 11063344 LP

Docente : Lic. Rosalia Lopez M.
Asignatura : PROGRAMACIÓN II (INF - 121) - Verano
Paralelo : “A”

La Paz, 28 de enero 2025

Elaboración del Proyecto de un Sistema para Gestión de Becas en el Contexto de la Programación Orientada a Objetos (POO)

Introducción

Muchas universidades enfrentan un problema importante relacionado con la gestión de becas, esto se debe a que una gran cantidad de estudiantes solicitan apoyo financiero y el personal administrativo tiene dificultades para manejar la gran cantidad de solicitudes, además, otro factor relevante es que muchos de estos estudiantes no siempre cumplen con los requisitos mínimos establecidos, lo que complica aún más el proceso y su correcta evaluación. Debido a esto se realiza un Sistema para la Gestión de Becas implementado en Java, integrando diagramas UML, constructores, sobrecarga de métodos, herencia, agregación, composición, genericidad, manejo de archivos, interfaces y patrones de diseño.

1. Definición del Proyecto

1.1 Descripción General

Este proyecto se centra en el desarrollo de un sistema para la gestión de becas, diseñado para facilitar diversos procesos relacionados a:

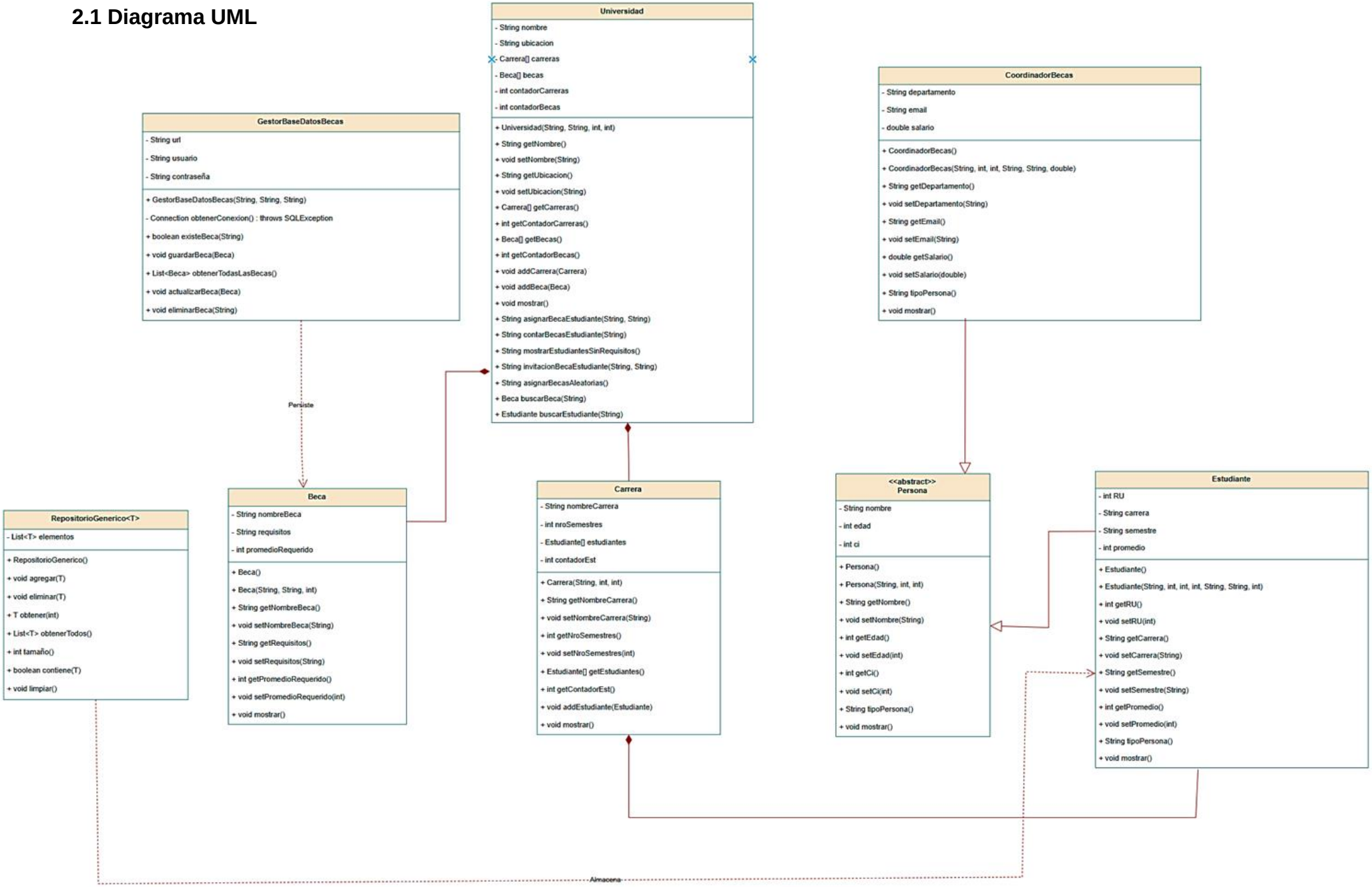
- Administración eficiente de los estudiantes beneficiados con becas.
- Identificación de aquellos que cumplen con los requisitos para acceder a una beca.
- Registro de estudiantes tanto con como sin beca.
- Gestión estable de las becas en función del desempeño académico reflejado en las notas de los estudiantes.

1.2 Objetivos

- Optimizar y facilitar la gestión de becas.
- Aplicar los conceptos fundamentales de POO.
- Obtener una copia de seguridad de los becados.
- Reducir el tiempo del proceso de asignación de becas.
- Sistematizar la gestión de becas.

2. Análisis y Diseño

2.1 Diagrama UML



2.2 Principios de Diseño

- ➔ Herencia: Las clases *CoordinadorBecas* y *Estudiante* heredan de *Persona*.
- ➔ Composición: La clase Universidad **contiene** al objeto de tipo Carrera y Beca, también Carrera alberga Estudiante.
- ➔ Agregación: El GestorBaseDatosBecas depende de la clase Beca para poder guardarlas, actualizarlas, etc. RepositorioGenerico es una clase genérica que maneja colecciones de cierto tipo (aquí se usa con Estudiante) y MainBecas "usa" muchas de las clases para instanciarlas y ejecutar métodos.
- ➔ Genericidad: Uso de colecciones genéricas para gestionar listas de estudiantes y becas.
- ➔ Interfaces: Implementación de una interfaz Gestionable (pantalla principal) para operaciones CRUD.
- ➔ Persistencia: En el RepositorioGenerico se encuentra el uso de la persistencia.
- ➔ Patrones de diseño: Estos patrones garantizan una experiencia de usuario optimizada y una gestión eficiente tanto para los solicitantes como para los administradores.

3. Implementación en Java

3.1 Estructura del Proyecto

Paquetes:

- LogicaBecasUniversitarias (donde se encuentran todas las clases y funciones necesarias para el sistema como ser: Beca, Carrera, Estudiante, etc).
- EntornoGrafico (se encuentran todos los códigos necesarios para que funcione el entorno gráfico, además sea accesible y cómodo para el usuario que tiene diferentes clases: a_inico, b_Becas, etc).
- Resources (es un paquete dedicado solamente a imágenes que permiten una mayor personalización).

3.2 Código Fuente

Clase base: Beca

```
package LogicaBecasUniversitarias;
public class Beca {
    private String nombreBeca;
    private String requisitos;
    private int promedioRequerido;
    public Beca() {
    }
    public Beca(String nombreBeca, String
requisitos, int promedioRequerido) {
        this.nombreBeca = nombreBeca;
        this.requisitos = requisitos;
        this.promedioRequerido = promedioRequerido;
    }
    public void mostrar() {
        System.out.println("\nBeca: " + nombreBeca
                + "\nRequisitos: " + requisitos
                + "\nPromedio requerido: " +
promedioRequerido);
    }    // Getters & Setters
    public String getNombreBeca() {
        return nombreBeca;
    }
    public void setNombreBeca(String nombreBeca) {
        this.nombreBeca = nombreBeca;
    }
    public String getRequisitos() {
        return requisitos;
    }
    public void setRequisitos(String requisitos) {
        this.requisitos = requisitos;
    }
    public int getPromedioRequerido() {
        return promedioRequerido;
    }
    public void setPromedioRequerido(int
promedioRequerido) {
        this.promedioRequerido = promedioRequerido;
    }
}
```

Clase base: Persona

```java
package LogicaBecasUniversitarias;
public abstract class Persona {
    protected String nombre;
    protected int edad;
    protected int ci;
    public Persona() {}
    public Persona(String nombre, int edad, int ci)
{
        this.nombre = nombre;
        this.edad = edad;
        this.ci = ci;
    }
    public abstract String tipoPersona();
    public void mostrar() {
        System.out.println("\nNombre: " + nombre
                + "\nEdad: " + edad
                + "\nCI: " + ci);
    }
    // Getters & Setters
public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    public int getCi() {
        return ci;
    }
    public void setCi(int ci) {
        this.ci = ci;
    }
}
```

Clase Derivada: Estudiante

```java
package LogicaBecasUniversitarias;
public class Estudiante extends Persona {
    private int RU;
    private String carrera;
    private String semestre;
    private int promedio;
    public Estudiante() {
        super();
    }
    public Estudiante(String nombre, int edad, int
ci, int RU,
            String carrera, String semestre, int
promedio) {
        super(nombre, edad, ci);
        this.RU = RU;
        this.carrera = carrera;
        this.semestre = semestre;
        this.promedio = promedio;
    }
    public String tipoPersona() {
        return "Estudiante";
    }
    public void mostrar() {
        System.out.println("\n=== Estudiante ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("CI: " + ci);
        System.out.println("RU: " + RU);
        System.out.println("Carrera: " + carrera);
        System.out.println("Semestre: " + semestre);
        System.out.println("Promedio: " + promedio);
    }
// Getters & Setters
    public int getRU() {
        return RU;
    }
    public void setRU(int RU) {
        this.RU = RU;
    }
    public String getCarrera() {
        return carrera;
    }
    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }
    public String getSemestre() {
        return semestre;
    }
    public void setSemestre(String semestre) {
        this.semestre = semestre;
    }
    public int getPromedio() {
        return promedio;
    }
    public void setPromedio(int promedio) {
        this.promedio = promedio;
    }
}
```

Clase Derivada: CoordinadorBecas

```
package LogicaBecasUniversitarias;
public class CoordinadorBecas extends Persona {
    private String departamento;
    private String email;
    private double salario;
    public CoordinadorBecas() {
        super();
    }
  public CoordinadorBecas(String nombre, int edad, int ci, String departamento,   String email, double salario) {
        super(nombre, edad, ci);
        this.departamento = departamento;
        this.email = email;
        this.salario = salario;
    }
    public String tipoPersona() {
        return "Coordinador de Becas";
    }
    public void mostrar() {
        super.mostrar();
        System.out.println("Departamento:     "     + departamento);
        System.out.println("Email: " + email);
        System.out.println("Salario: " + salario);
    } // Getters and Setters
    public String getDepartamento() {
        return departamento;
    }
    public void setDepartamento(String departamento) {
        this.departamento = departamento;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public double getSalario() {
        return salario;
    }
    public void setSalario(double salario) {
        this.salario = salario;
    }
}
```

Clase Controlador: GestorBaseDatosBecas

```
package LogicaBecasUniversitarias;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.sql.SQLException;
public class GestorBaseDatosBecas {
    private String url;
    private String usuario;
    private String contraseña;
    public GestorBaseDatosBecas(String url, String usuario, String contraseña) {
        this.url = url;
        this.usuario = usuario;
        this.contraseña = contraseña;
    }
    // Establecer conexión a la base de datos
    private Connection obtenerConexion() throws SQLException {
        return      DriverManager.getConnection(url, usuario, contraseña);
    }
    public boolean existeBeca(String nombreBeca) {
        String sql = "SELECT COUNT(*) FROM becas WHERE nombre_beca = ?";
        try (Connection conn = obtenerConexion(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return rs.getInt(1) > 0;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }
    // Método para guardar una beca en la base de datos
    public void guardarBeca(Beca beca) {
        if (existeBeca(beca.getNombreBeca())) {
            System.out.println("La    beca    \""    + beca.getNombreBeca() + "\" ya existe en la base de datos.");
            return;
        }
        String sql = "INSERT INTO becas (nombre_beca, requisitos, promedio_requerido) VALUES (?, ?, ?)";
        try (Connection conn = obtenerConexion(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, beca.getNombreBeca());
            pstmt.setString(2, beca.getRequisitos());
            pstmt.setInt(3,
```

```
beca.getPromedioRequerido());
            pstmt.executeUpdate();
           System.out.println("Beca       \""       +
beca.getNombreBeca() + "\" guardada exitosamente.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Método para recuperar todas las becas de la
base de datos
    public List<Beca> obtenerTodasLasBecas() {
        List<Beca> becas = new ArrayList<>();
        String sql = "SELECT nombre_beca, requisitos,
promedio_requerido FROM becas";
        try (Connection conn = obtenerConexion();
Statement stmt = conn.createStatement(); ResultSet
rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Beca beca = new Beca(

rs.getString("nombre_beca"),
                        rs.getString("requisitos"),

rs.getInt("promedio_requerido")
                );
                becas.add(beca);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return becas;
    }
```

```
    // Método para actualizar una beca existente
    public void actualizarBeca(Beca beca) {
        String sql = "UPDATE becas SET requisitos =
?, promedio_requerido = ? WHERE nombre_beca = ?";
        try (Connection conn = obtenerConexion();
PreparedStatement              pstmt              =
conn.prepareStatement(sql)) {
            pstmt.setString(1,
beca.getRequisitos());
            pstmt.setInt(2,
beca.getPromedioRequerido());
            pstmt.setString(3,
beca.getNombreBeca());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Método para eliminar una beca
    public void eliminarBeca(String nombreBeca) {
        String sql = "DELETE FROM becas WHERE
nombre_beca = ?";
        try (Connection conn = obtenerConexion();
PreparedStatement              pstmt              =
conn.prepareStatement(sql)) {
            pstmt.setString(1, nombreBeca);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Clase Controlador: Carrera

```
package LogicaBecasUniversitarias;
public class Carrera {
    private String nombreCarrera;
    private int nroSemestres;
    private Estudiante[] estudiantes;
    private int contadorEst;
    public    Carrera(String    nombreCarrera,    int
nroSemestres, int maxEstudiantes) {
        this.nombreCarrera = nombreCarrera;
        this.nroSemestres = nroSemestres;
        this.estudiantes            =            new
Estudiante[maxEstudiantes];
        this.contadorEst = 0;
    }
    // Agregar un estudiante al arreglo
    public void addEstudiante(Estudiante e) {
        if (contadorEst < estudiantes.length) {
            estudiantes[contadorEst] = e;
            contadorEst++;
        } else {
            System.out.println("No se puede agregar
más estudiantes en " + nombreCarrera);
```

```
    }
    public void mostrar() {
        System.out.println("\n-----  CARRERA:  "  +
nombreCarrera + " -----");
        System.out.println("Semestres:      "      +
nroSemestres);
        System.out.println("Estudiantes:");
        for (int i = 0; i < contadorEst; i++) {
            estudiantes[i].mostrar();
        }
    }
    // Getters & Setters
    public String getNombreCarrera() {
        return nombreCarrera;
    }
    public       void       setNombreCarrera(String
nombreCarrera) {
        this.nombreCarrera = nombreCarrera;
    }
    public int getNroSemestres() {
        return nroSemestres;
    }
```

```
    public void setNroSemestres(int nroSemestres) {
        this.nroSemestres = nroSemestres;
    }
    public Estudiante[] getEstudiantes() {
        return estudiantes;
    }
    public int getContadorEst() {
        return contadorEst;
    }
}
```

Clase Controlador: Universidad

```
    package LogicaBecasUniversitarias;
import java.util.Random;
public class Universidad {
    private String nombre;
    private String ubicacion;
    private Carrera[] carreras;
    private int contadorCarreras;
    private Beca[] becas = new Beca[10];
    private int contadorBecas;
    public Universidad(String nombre, String
ubicacion, int maxCarreras, int maxBecas) {
        this.nombre = nombre;
        this.ubicacion = ubicacion;
        carreras = new Carrera[maxCarreras];
        becas = new Beca[maxBecas];
        contadorCarreras = 0;
        contadorBecas = 0;
    }
    // Método para agregar una carrera al arreglo
de carreras
    public void addCarrera(Carrera c) {
        if (contadorCarreras < carreras.length) {
            carreras[contadorCarreras] = c;
            contadorCarreras++;
            System.out.println("No se pueden
agregar más carreras.");
        }
    }
    // Método para agregar una beca al arreglo de
becas
    public void addBeca(Beca b) {
        if (contadorBecas == becas.length) {
            Beca[] x = new Beca[becas.length * 2];
            for (int i = 0; i < becas.length; i++)
{
                x[i] = becas[i];
            }
            becas = x;
        }
        becas[contadorBecas] = b;
        contadorBecas++;
    }
```

```
    // Método para mostrar la información de la
universidad, sus becas y carreras
    public void mostrar() {

System.out.println("\n============================
==");
        System.out.println("UNIVERSIDAD: " +
nombre);          System.out.println("UBICACIÓN: "
+ ubicacion);
        System.out.println("\n-- BECAS DISPONIBLES
--");
        for (int i = 0; i < contadorBecas; i++) {
            becas[i].mostrar();
        }

        System.out.println("\n-- CARRERAS --");
        for (int i = 0; i < contadorCarreras; i++)
{
            carreras[i].mostrar();
        }

System.out.println("==============================
\n");
    }

    // Método para asignar una beca a un estudiante
específico
    public void asignarBecaEstudiante(String
nombreEst, String nombreBeca) {
        Beca laBeca = buscarBeca(nombreBeca);
        if (laBeca == null) {
            System.out.println("No se encontró la
Beca " + nombreBeca);
            return;
        }
        Estudiante est =
buscarEstudiante(nombreEst);
        if (est == null) {
            System.out.println("No se encontró al
estudiante " + nombreEst);
            return;
        }
```

```
        if (est.getPromedio() >=
laBeca.getPromedioRequerido()) {
            System.out.println("El estudiante " +
est.getNombre()
                    + " (promedio " +
est.getPromedio()
                    + ") recibe la beca " +
laBeca.getNombreBeca()
                    + " (req. " +
laBeca.getPromedioRequerido() +
        } else {
            System.out.println("El estudiante " +
est.getNombre()
                    + " (promedio " +
est.getPromedio()
                    + ") NO cumple para la beca " +
laBeca.getNombreBeca()
                    + " (req. " +
laBeca.getPromedioRequerido() + ")");
        }
    }
    // Método para contar y mostrar las becas que
un estudiante puede adquirir
    public void contarBecasEstudiante(String
nombreEst) {
        Estudiante e = buscarEstudiante(nombreEst);
        if (e == null) {
            System.out.println("No existe el
estudiante " + nombreEst);
            return;
        }
        int contador = 0;
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < contadorBecas; i++) {
            Beca b = becas[i];
            if (e.getPromedio() >=
b.getPromedioRequerido()) {
                contador++;
                sb.append(" -
").append(b.getNombreBeca())
                        .append(" (req:
").append(b.getPromedioRequerido()).append(")\n");
            }
        }
        if (contador == 0) {
            System.out.println("Estudiante " +
e.getNombre()
                    + " no puede adquirir ninguna
beca (promedio " + e.getPromedio() + ")");
        } else {
            System.out.println("Estudiante " +
e.getNombre()
                    + " (promedio " +
e.getPromedio()
                    + ") puede adquirir " +
contador + " becas:\n" + sb.toString());
        }
    }
    // Método para mostrar estudiantes que no
cumplen los requisitos de beca (promedio <= 69)
    public void mostrarEstudiantesSinRequisitos() {
        System.out.println("\n** Estudiantes con
promedio <= 69 **");
        int total = 0;
        for (int i = 0; i < contadorCarreras; i++)
{
            Carrera c = carreras[i];
            Estudiante[] arr = c.getEstudiantes();
            for (int j = 0; j < c.getContadorEst();
j++) {
                Estudiante est = arr[j];
                if (est.getPromedio() <= 69) {
                    total++;
                    System.out.println("- " +
est.getNombre()
                            + " (promedio: " +
est.getPromedio() + "), carrera: "
                            +
c.getNombreCarrera());
                }
            }
        }
        if (total == 0) {
            System.out.println("No hay estudiantes
con promedio <= 69.");
        }
    }
    // Método para invitar a un estudiante a una
beca por su desempeño
    public void invitacionBecaEstudiante(String
nombreEst, String nombreBeca) {
        Estudiante e = buscarEstudiante(nombreEst);
        if (e == null) {
            System.out.println("No existe el
estudiante " + nombreEst);
            return;
        }
        Beca b = buscarBeca(nombreBeca);
        if (b == null) {
            System.out.println("No existe la beca "
+ nombreBeca);
            Return;
        }
        if (e.getPromedio() >=
b.getPromedioRequerido()) {
            System.out.println("INVITACIÓN: El
estudiante " + e.getNombre()
                    + " de la carrera " +
e.getCarrera()
                    + " con promedio " +
e.getPromedio()
                    + " es invitado a la beca " +
b.getNombreBeca()
                    + " (req. " +
b.getPromedioRequerido() + ")");
        } else {
            System.out.println("El estudiante " +
e.getNombre()
                    + " no cumple para la
invitación a la beca " + b.getNombreBeca());
```

```
    // Método para asignar becas aleatorias a estudiantes que cumplen los requisitos
    public void asignarBecasAleatorias() {
        System.out.println("\n*** Asignar becas aleatorias ***");
        Random random = new Random();
        int totalAsig = 0;
        for (int i = 0; i < contadorCarreras; i++) {
            Carrera c = carreras[i];
            Estudiante[] arr = c.getEstudiantes();
            for (int j = 0; j < c.getContadorEst(); j++) {
                Estudiante e = arr[j];
                if (e.getPromedio() >= 70 && contadorBecas > 0) {
                    int idx = random.nextInt(contadorBecas);
                    Beca b = becas[idx];
                        totalAsig++;
                        System.out.println("El estudiante " + e.getNombre()
                                + " (" + e.getCarrera() + ", prom: " + e.getPromedio()+ ") recibe beca aleatoria -> " + b.getNombreBeca());
                    }
                }
            }
        System.out.println("Total de becas aleatorias asignadas: " + totalAsig);
    }
    // Método auxiliar para buscar una beca por nombre
    public Beca buscarBeca(String nombreBeca) {
        for (int i = 0; i < contadorBecas; i++) {
            if (becas[i].getNombreBeca().equalsIgnoreCase(nombreBeca)) { //
                return becas[i];
            }
        return null;
    }
    // Método auxiliar para buscar un estudiante por nombre
    public Estudiante buscarEstudiante(String nombreEst) {
        for (int i = 0; i < contadorCarreras; i++) {
            Carrera c = carreras[i];
            Estudiante[] e = c.getEstudiantes();
            for (int j = 0; j < c.getContadorEst(); j++) {
                if (e[j].getNombre().equalsIgnoreCase(nombreEst)) {
                    return e[j];
                }
            }
        }
        return null;
    }

    // Getters y Setters
    public String getNombre() {
        return nombre;
    }

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

    public String getUbicacion() {
        return ubicacion;
    }

    public void setUbicacion(String ubicacion) {
        this.ubicacion = ubicacion;
    }

    public Carrera[] getCarreras() {
        return carreras;
    }

    public int getContadorCarreras() {
        return contadorCarreras;
    }

    public Beca[] getBecas() {
        return becas;
    }

    public int getContadorBecas() {
        return contadorBecas;
    }
}
```

Clase Controlador: RepositorioGenerico

```
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
public class RepositorioGenerico<T> {
    private List<T> elementos;
    public RepositorioGenerico() {
        this.elementos = new ArrayList<>();
    }
    public void agregar(T elemento) {
        elementos.add(elemento);
    }
    public void eliminar(T elemento) {
        elementos.remove(elemento);
    }
    public T obtener(int indice) {
        return elementos.get(indice);
    }
    public List<T> obtenerTodos() {
        return new ArrayList<>(elementos);
    }
    public int tamaño() {
        return elementos.size();
    }
    public boolean contiene(T elemento) {
        return elementos.contains(elemento);
    }
    public void limpiar() {
        elementos.clear();
    }
    public void mostrar() {
        for (T elemento : elementos) {
            try {
                Method metodoMostrar = elemento.getClass().getMethod("mostrar");
                metodoMostrar.invoke(elemento);
            } catch (Exception e) {
                System.out.println("El objeto no tiene un método 'mostrar': " + elemento);
            }
        }
    }
    public List<T> getElementos() {
        return elementos;
    }
    public void setElementos(List<T> elementos) {
        this.elementos = elementos;
    }
}
```

Clase Main:

```
package LogicaBecasUniversitarias;
public class MainBecas {
    public static void main(String[] args) {
        // Crear Universidad
        Universidad universidad = new
Universidad("Escuela Militar de Ingenieria",
"Rafael Pabon - Irpavi", 10, 10);
        Carrera car1 = new Carrera("Ingeniería de
Sistemas", 7, 100);
        universidad.addCarrera(car1);
        Carrera car2 = new Carrera("Ingeniería de
Software", 8, 100);
        universidad.addCarrera(car2);
        Carrera car3 = new Carrera("Ingeniería
Electrónica", 9, 100);
        universidad.addCarrera(car3);
        Carrera car4 = new Carrera("Ingeniería
Mecánica", 7, 100);
        universidad.addCarrera(car4);
        Carrera car5 = new Carrera("Ingeniería
Civil", 6, 100);
        universidad.addCarrera(car5);
        Carrera car6 = new Carrera("Ingeniería
Industrial", 5, 100);
        universidad.addCarrera(car6);
        Carrera car7 = new Carrera("Ingeniería
Ambiental", 4, 100);
        universidad.addCarrera(car7);
        Carrera car8 = new Carrera("Ingeniería en
Telecomunicaciones", 8, 100);
        universidad.addCarrera(car8);
        Carrera car9 = new Carrera("Ingeniería en
Transporte", 8, 100);
        universidad.addCarrera(car9);
        Carrera car10 = new Carrera("Ingeniería en
Energías Renovables", 8, 100);
        universidad.addCarrera(car10);
        // Crear Estudiantes
        //1.Estudiantes Ing de Sistemas (6)
        Estudiante e1 = new Estudiante("Carlos
Pérez", 20, 12345, 678901, "Ingeniería de
Sistemas", "2do", 92);
        Estudiante e2 = new Estudiante("Ana Gómez",
21, 54321, 123456, "Ingeniería de Sistemas", "2do",
89);
        Estudiante e3 = new Estudiante("Luis
Fernández", 22, 98765, 654321, "Ingeniería de
Sistemas", "3er", 100);
        Estudiante e4 = new Estudiante("María
López", 19, 67890, 789012, "Ingeniería de
Sistemas", "3er", 70);
        Estudiante e5 = new Estudiante("Pedro
Martínez", 23, 13579, 135790, "Ingeniería de
Sistemas", "4to", 69);
        Estudiante e6 = new Estudiante("Sofía
Morales", 20, 24680, 246801, "Ingeniería de
Sistemas", "4to", 60);
        //2.Estudiantes Ing Software (7)
        Estudiante e7 = new Estudiante("Javier
Torres", 21, 75319, 753190, "Ingeniería de
Software", "2do", 94);
        Estudiante e8 = new Estudiante("Lucía
Ramírez", 22, 86420, 864201, "Ingeniería de
Software", "2do", 78);
        Estudiante e9 = new Estudiante("Mario
Ríos", 23, 97531, 975310, "Ingeniería de Software",
"5to", 90);
        Estudiante e10 = new Estudiante("Manuel
Fernandez", 22, 24612, 246802, "Ingeniería de
Software", "3ro", 88);
        Estudiante e11 = new Estudiante("Andres
Casas", 25, 25356, 24655, "Ingeniería de Software",
"4to", 70);
        Estudiante e12 = new Estudiante("Bismarck
Valencia", 24, 24681, 244802, "Ingeniería de
Software", "2ro", 65);
        Estudiante e13 = new Estudiante("Joel
Saavedra", 21, 27542, 246812, "Ingeniería de
Software", "1ro", 70);
        //3.Estudiantes ing Electronica (3)
        Estudiante e14 = new Estudiante("Ricardo
Leon", 20, 24681, 246802, "Ingeniería Electrónica",
"6to", 78);
        Estudiante e15 = new Estudiante("Joel
Quispe", 22, 34752, 259667, "Ingeniería
Electrónica", "3ro", 95);
        Estudiante e16 = new Estudiante("Emmanuel
Llanos", 24, 55532, 789342, "Ingeniería
Electrónica", "1ro", 100);
        //4.Estudiantes Ing Mecánica (4)
        Estudiante e17 = new Estudiante("Leslie
Cabrera", 22, 55501, 700123, "Ingeniería Mecánica",
"5to", 91);
        Estudiante e18 = new Estudiante("Fernando
Ortega", 21, 55546, 712354, "Ingeniería Mecánica",
"4to", 87);
        Estudiante e19 = new Estudiante("Marcos
Fernandez", 18, 51235, 723543, "Ingeniería
Mecánica", "1ro", 60);
        Estudiante e20 = new Estudiante("Julio
Mayta", 20, 19284, 712389, "Ingeniería Mecánica",
"3ro", 70);
        //5.Estudiantes Ing Civil (6)
        Estudiante e21 = new Estudiante("Daniel
Iriarte", 30, 55502, 700124, "Ingeniería Civil",
"7to", 87);
        Estudiante e22 = new Estudiante("Rolando
Crespo", 32, 24865, 445258, "Ingeniería Civil",
"9to", 60);
        Estudiante e23 = new Estudiante("Jayce
```

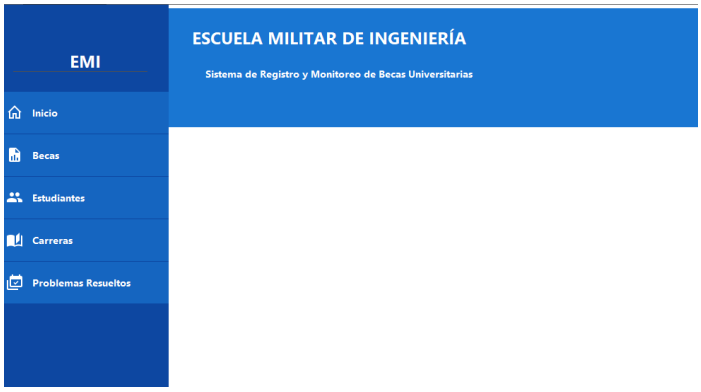
```
Morales", 20, 24680, 246801, "Ingeniería de
Sistemas", "4to", 60);
        //2.Estudiantes Ing Software (7)
        Estudiante e7 = new Estudiante("Javier
Torres", 21, 75319, 753190, "Ingeniería de
Software", "2do", 94);
        Estudiante e8 = new Estudiante("Lucía
Ramírez", 22, 86420, 864201, "Ingeniería de
Software", "2do", 78);
        Estudiante e9 = new Estudiante("Mario
Ríos", 23, 97531, 975310, "Ingeniería de Software",
"5to", 90);
        Estudiante e10 = new Estudiante("Manuel
Fernandez", 22, 24612, 246802, "Ingeniería de
Software", "3ro", 88);
        Estudiante e11 = new Estudiante("Andres
Casas", 25, 25356, 24655, "Ingeniería de Software",
"4to", 70);
        Estudiante e12 = new Estudiante("Bismarck
Valencia", 24, 24681, 244802, "Ingeniería de
Software", "2ro", 65);
        Estudiante e13 = new Estudiante("Joel
Saavedra", 21, 27542, 246812, "Ingeniería de
Software", "1ro", 70);
        //3.Estudiantes ing Electronica (3)
        Estudiante e14 = new Estudiante("Ricardo
Leon", 20, 24681, 246802, "Ingeniería Electrónica",
"6to", 78);
        Estudiante e15 = new Estudiante("Joel
Quispe", 22, 34752, 259667, "Ingeniería
Electrónica", "3ro", 95);
        Estudiante e16 = new Estudiante("Emmanuel
Llanos", 24, 55532, 789342, "Ingeniería
Electrónica", "1ro", 100);
        //4.Estudiantes Ing Mecánica (4)
        Estudiante e17 = new Estudiante("Leslie
Cabrera", 22, 55501, 700123, "Ingeniería Mecánica",
"5to", 91);
        Estudiante e18 = new Estudiante("Fernando
Ortega", 21, 55546, 712354, "Ingeniería Mecánica",
"4to", 87);
        Estudiante e19 = new Estudiante("Marcos
Fernandez", 18, 51235, 723543, "Ingeniería
Mecánica", "1ro", 60);
        Estudiante e20 = new Estudiante("Julio
Mayta", 20, 19284, 712389, "Ingeniería Mecánica",
"3ro", 70);
        //5.Estudiantes Ing Civil (6)
        Estudiante e21 = new Estudiante("Daniel
Iriarte", 30, 55502, 700124, "Ingeniería Civil",
"7to", 87);
        Estudiante e22 = new Estudiante("Rolando
Crespo", 32, 24865, 445258, "Ingeniería Civil",
"9to", 60);
        Estudiante e23 = new Estudiante("Jayce
Calisalla", 19, 55502, 741963, "Ingeniería Civil",
"2do", 66);
        Estudiante e24 = new Estudiante("Andres
Mayta", 25, 21367, 159753, "Ingeniería Civil",
"5to", 85);
        Estudiante e25 = new Estudiante("Alejandro
Fernandez", 21, 79513, 753258, "Ingeniería Civil",
"2to", 77);
        Estudiante e26 = new Estudiante("Marcelo
Justiniano", 20, 47512, 953751, "Ingeniería Civil",
"4to", 99);
        //6.Estudiantes Ing Industrial (6)
        Estudiante e27 = new Estudiante("Alejandra
Cabrera", 21, 48624, 486159, "Ingeniería
Industrial", "2to", 87);
        Estudiante e28 = new Estudiante("Emily
Rose", 23, 22436, 113465, "Ingeniería Industrial",
"4to", 70);
        Estudiante e29 = new Estudiante("Adriana
Conde", 25, 48612, 765622, "Ingeniería Industrial",
"6to", 77);
        Estudiante e30 = new Estudiante("Erick
Fernandez", 27, 95632, 700124, "Ingeniería
Industrial", "8to", 67);
        Estudiante e31 = new Estudiante("Alejandro
Uzquiano", 29, 71233, 485225, "Ingeniería
Industrial", "2to", 90);
        Estudiante e32 = new Estudiante("Daniel
Moscoso", 30, 75649, 996553, "Ingeniería
Industrial", "1to", 92);
        //7.Estudiantes Ing Ambiental (3)
        Estudiante e33 = new Estudiante("Mariana
Fernandez", 21, 55463, 448526, "Ingeniería
Ambiental", "5to", 100);
        Estudiante e34 = new Estudiante("Natalia
Tejerina", 21, 48655, 664645, "Ingeniería
Ambiental", "5to", 60);
        Estudiante e35 = new Estudiante("Alejandro
Fernandez", 21, 11243, 123455, "Ingeniería
Ambiental", "5to", 87);
        //8.Estudiantes Ing en Telecomunicaciones
(4)
        Estudiante e36 = new Estudiante("Omar
Saavedra", 24, 44525, 700124, "Ingeniería en
Telecomunicaciones", "4to", 85);
        Estudiante e37 = new Estudiante("Fernando
Torrez", 22, 48852, 333555, "Ingeniería en
Telecomunicaciones", "2do", 77);
        Estudiante e38 = new Estudiante("Sthepanie
Linarez", 21, 44525, 486122, "Ingeniería en
Telecomunicaciones", "3to", 84);
        Estudiante e39 = new Estudiante("Alexys
Camara", 18, 99653, 996554, "Ingeniería en
Telecomunicaciones", "1ro", 81);
        //9.Estudiantes Ing en Transporte (3)
        Estudiante e40 = new Estudiante("Alejandra
Fernandez", 24, 44566, 700124, "Ingeniería en
Transporte", "7mo", 70);
        Estudiante e41 = new Estudiante("Ana Leon",
25, 11258, 456983, "Ingeniería en Transporte",
"2to", 90);
        Estudiante e42 = new Estudiante("Augusto
Choque", 27, 24475, 885221, "Ingeniería en
Transporte", "5to", 60);
        //10.Estudiantes Ing en Energias Renovables
(2)
        Estudiante e43 = new Estudiante("Rogelio
```

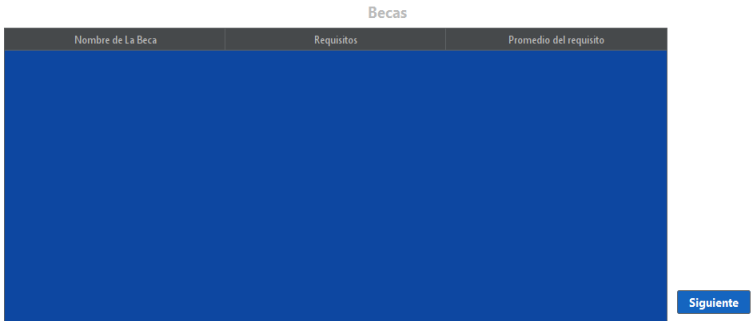
```
Mamani", 21, 12345, 554394, "Ingeniería en Energías
Renovables", "5to", 84);
        Estudiante e44 = new Estudiante("Marcelo
Aruquipa", 21, 54321, 663698, "Ingeniería en
Energías Renovables", "5to", 100);
        // Agregar Estudiantes a Carreras
        //1. Ing de Sistemas
        car1.addEstudiante(e1);
        car1.addEstudiante(e2);
        car1.addEstudiante(e3);
        car1.addEstudiante(e4);
        car1.addEstudiante(e5);
        car1.addEstudiante(e6);
        //2. Ing de Software
        car2.addEstudiante(e7);
        car2.addEstudiante(e8);
        car2.addEstudiante(e9);
        car2.addEstudiante(e10);
        car2.addEstudiante(e11);
        car2.addEstudiante(e12);
        car2.addEstudiante(e13);
        //3. Ing Electronica
        car3.addEstudiante(e14);
        car3.addEstudiante(e15);
        car3.addEstudiante(e16);
        //4. Ing Mecánica
        car4.addEstudiante(e17);
        car4.addEstudiante(e18);
        car4.addEstudiante(e19);
        car4.addEstudiante(e20);
        //5. Ing Civil
        car5.addEstudiante(e21);
        car5.addEstudiante(e22);
        car5.addEstudiante(e23);
        car5.addEstudiante(e24);
        car5.addEstudiante(e25);
        car5.addEstudiante(e26);
        //6. Ing Industrial
        car6.addEstudiante(e27);
        car6.addEstudiante(e28);
        car6.addEstudiante(e29);
        car6.addEstudiante(e30);
        car6.addEstudiante(e31);
        car6.addEstudiante(e32);
        //7. Ing Ambiental
        car7.addEstudiante(e33);
        car7.addEstudiante(e34);
        car7.addEstudiante(e35);
        //8. Ing en Telecomunicaciones
        car8.addEstudiante(e36);
        car8.addEstudiante(e37);
        car8.addEstudiante(e38);
        car8.addEstudiante(e39);
        //9. Ing en Transporte
        car9.addEstudiante(e40);
        car9.addEstudiante(e41);
        car9.addEstudiante(e42);
        //10. Ing en Energias Renovables
        car10.addEstudiante(e43);
        car10.addEstudiante(e44);
        // Crear Becas
```

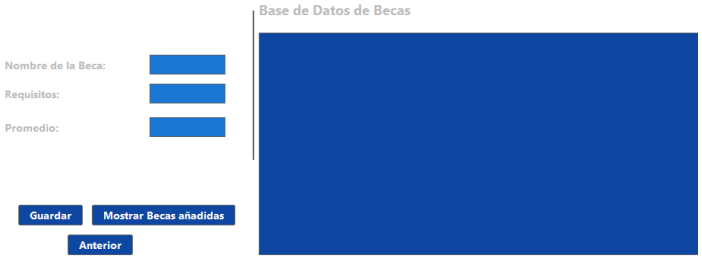
```
Beca de Excelencia", "Promedio mayor a 90", 92);
        Beca b2 = new Beca("Beca Deportiva", "Ser
parte del equipo de fútbol", 70);
        Beca b3 = new Beca("Beca de Investigación",
"Participar en un proyecto", 88);
        Beca b4 = new Beca("Beca Cultural",
"Participar en actividades culturales", 80);
        Beca b5 = new Beca("Beca de Honor",
"Estudiantes destacados en su área", 100);
        Beca b6 = new Beca("Beca de Liderazgo",
"Demostrar habilidades de liderazgo", 87);
        Beca b7 = new Beca("Beca de Servicio
Comunitario", "Voluntariado comprobado", 82);
        Beca b8 = new Beca("Beca Técnica", "Tener
habilidades técnicas", 75);
        Beca b9 = new Beca("Beca de Innovación",
"Proyectos innovadores", 85);
        Beca b10 = new Beca("Beca de Desempeño
Académico", "Mantener un promedio mínimo", 84);
        universidad.addBeca(b1);
        universidad.addBeca(b2);
        universidad.addBeca(b3);
        universidad.addBeca(b4);
        universidad.addBeca(b5);
        universidad.addBeca(b6);
        universidad.addBeca(b7);
        universidad.addBeca(b8);
        universidad.addBeca(b9);
        universidad.addBeca(b10);
        // Usar Repositorio Genérico
        RepositorioGenerico<Beca> repositorioBecas
= new RepositorioGenerico<>();
        repositorioBecas.agregar(b1);
        repositorioBecas.agregar(b2);
        repositorioBecas.agregar(b3);
        repositorioBecas.agregar(b4);
        repositorioBecas.agregar(b5);
        repositorioBecas.agregar(b6);
        repositorioBecas.agregar(b7);
        repositorioBecas.agregar(b8);
        repositorioBecas.agregar(b9);
        repositorioBecas.agregar(b10);
// Crear Coordinador de Becas
        CoordinadorBecas coordinador = new
CoordinadorBecas(
                "Carlos Rodríguez", 45, 54321,
                "Departamento de Becas",
"carlos.rodriguez@universidad.edu", 5000.0
        );
        // Configurar Gestor de Base de Datos
(ejemplo de conexión)
        GestorBaseDatosBecas gestorBD = new
GestorBaseDatosBecas(

"jdbc:mysql://sql10.freesqldatabase.com:3306/sql107
59513",
                "sql10759513",
                "nK6aMYm38S"
        );
        // Guardar becas en la base de datos
        gestorBD.guardarBeca(b1);
```

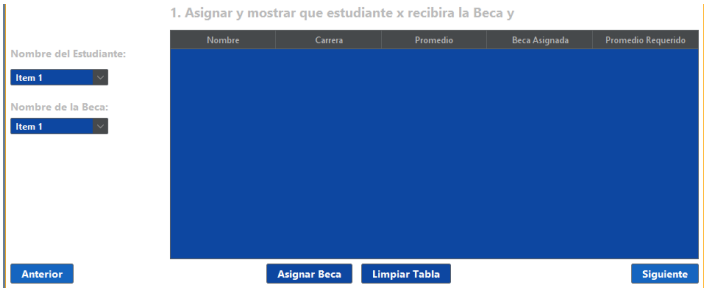
```
        gestorBD.guardarBeca(b2);
        gestorBD.guardarBeca(b3);
        gestorBD.guardarBeca(b4);
        gestorBD.guardarBeca(b5);
        gestorBD.guardarBeca(b6);
        gestorBD.guardarBeca(b7);
        gestorBD.guardarBeca(b8);
        gestorBD.guardarBeca(b9);
        gestorBD.guardarBeca(b10);
        universidad.mostrar();
        coordinador.mostrar();
        System.out.println(" ");
        System.out.println("Repositorio generico de Becas: ");
        repositorioBecas.mostrar();
        System.out.println(" ");
        // LLAMAR a cada uno de los 5 problemas usando métodos en la clase Universidad:
        // 1) Asignar y mostrar que estudiante X recibirá la Beca Y
        System.out.println("1) Asignar Beca:");
        universidad.asignarBecaEstudiante("Carlos Pérez", "Beca de Excelencia");
        // 2) Contar y Mostrar las becas que puede adquirir el estudiante X
        System.out.println("\n2) Contar becas posibles para un estudiante:");
        universidad.contarBecasEstudiante("Carlos Pérez");
        // 3) Mostrar a todos los estudiantes que no cumplan los requisitos
        System.out.println("\n3) Estudiantes que no cumplen los requisitos de beca");
        universidad.mostrarEstudiantesSinRequisitos();
        // 4) Invitación a la beca por desempeño
        System.out.println("\n4) Invitación a Beca:");
        universidad.invitacionBecaEstudiante("Luis Fernández", "Beca de Honor");
        // 5) Asignar becas aleatorias a los estudiantes que cumplan los requisitos
        System.out.println("\n5) Asignar becas aleatorias:");
        universidad.asignarBecasAleatorias();
    }
}
```

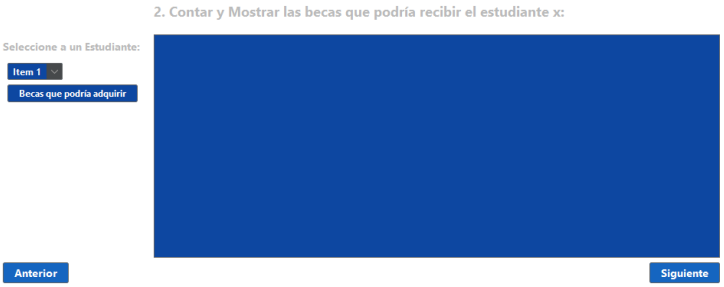
3.3 Diseño de Interfaces

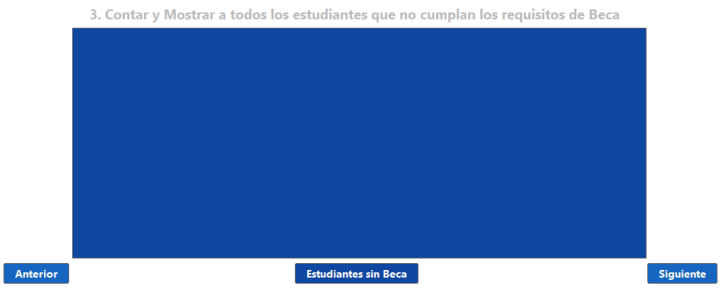


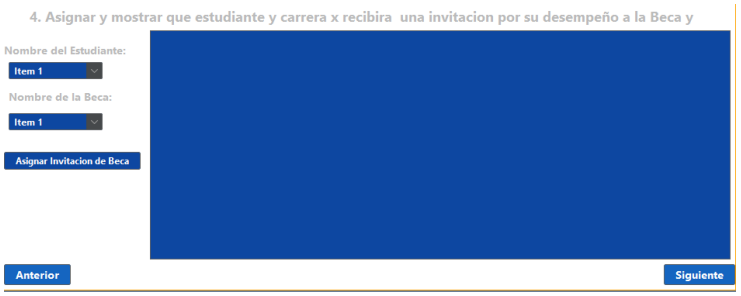


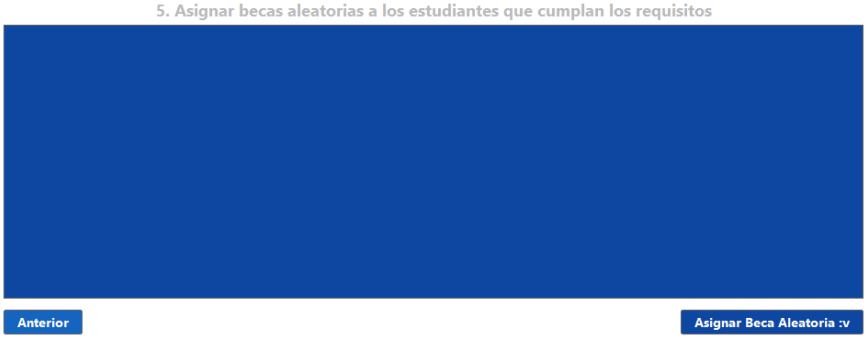












3.4 Manejo de Archivos

Cuando un estudiante solicita una beca, sus datos (nombre, correo, promedio, etc.) se **persisten** en la base de datos MySQL, de esta manera los administradores pueden conectarse al sistema para ver estas solicitudes almacenadas, analizarlas o modificarlas según sea necesario.

```
// Establecer conexión a la base de datos
    private Connection obtenerConexion() throws SQLException {
        return DriverManager.getConnection(url, usuario, contraseña);
    }

    public boolean existeBeca(String nombreBeca) {
        String sql = "SELECT COUNT(*) FROM becas WHERE nombre_beca = ?";
        try (Connection conn = obtenerConexion(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {

            pstmt.setString(1, nombreBeca);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return rs.getInt(1) > 0;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }
```

3.5 Uso de Patrones de Diseño

- **Encabezado con un llamado a la acción:** Incluye un título destacado como *"Escuela militar de ingeniería"*.
- **Barra de navegación:** Acceso rápido a secciones clave como "Inicio", "Solicitar Beca", "Información de Requisitos", "Contacto", y "Acceso para Administradores".
- **Formulario dinámico:** Permite solicitar una beca, adaptándose según los requisitos específicos de cada programa.
- **Seguimiento de solicitudes:** Una barra de progreso que indique el estado de la solicitud (pendiente, en revisión, aprobado, rechazado).
- **Evaluación automática:** Un algoritmo que evalúe requisitos mínimos como notas o ingresos.
- **Lista de becas disponibles:** Detalladas con descripciones, requisitos, beneficios y fechas límite.
- **Filtros de búsqueda:** Categorías como "becas por mérito académico", "becas económicas", "internacionales", etc.

4. Pruebas del Sistema

Página principal

Página de Becas

Tabla de todos los estudiantes

Asignación de Beca Y al estudiante X

Becas que un estudiante puede recibir

Tabla de estudiantes sin Beca

Página para asignación de una Beca

Becas aleatorias para estudiantes que puede obtener

5. Conclusión

El proyecto tiene como objetivo transformar por completo la Gestión de Becas, logrando que este proceso se lleve a cabo de una manera más eficaz y eficiente. Esto se consigue a través de la aplicación de la Programación Orientada a Objetos (POO), permitiendo manejar de forma estructurada y efectiva elementos clave como los estudiantes, las becas, las carreras, entre otros. Además, el sistema implementado facilita notablemente su uso mediante la integración estratégica de patrones de diseño. Todo esto no solo mejora la funcionalidad y accesibilidad del sistema, sino que también garantiza que se puedan cumplir de manera efectiva los objetivos planteados desde el principio del proyecto, optimizando los procesos y beneficiando tanto a los administradores como a los solicitantes.