**Universidad Veracruzana**

Facultad de Estadística e Informática

Licenciatura en Ingeniería de Software

Documento de Especificación de Requerimientos

Sistema “Ared Espacio”

90% de avance del Proyecto

Integrantes del equipo

GARCIA DELGADO JONATHAN

JUAREZ CAPISTRAN MAURICIO

CORTINA GAMBOA EHECATL SAID

Contenido

[Descripción de los casos de uso Implementados 4](#_Toc482839717)

[Diagramas de Robustez 14](#_Toc482839718)

[CU-10. Pagar Maestro 14](#_Toc482839719)

[CU-11. Registrar Egreso 14](#_Toc482839720)

[CU-12. Ver historial de Egresos 15](#_Toc482839721)

[CU-13. Ver historial de Ingresos 15](#_Toc482839722)

[CU-14. Ver Notificaciones 16](#_Toc482839723)

[CU-15. Crear Promoción 16](#_Toc482839724)

[CU-16. Modificar Promoción 17](#_Toc482839725)

[CU-17. Eliminar Promoción 17](#_Toc482839726)

[CU-18. Registrar Asistencia 18](#_Toc482839727)

[CU-19. Cambiar Alumno de Grupo 18](#_Toc482839728)

[Modelo de Dominio 19](#_Toc482839729)

[Diagrama de Clases 20](#_Toc482839730)

[Diagramas de Secuencia 21](#_Toc482839731)

[CU-10. Pagar Maestro 21](#_Toc482839732)

[CU-11. Registrar Egreso 22](#_Toc482839733)

[CU-12. Ver historial de Egresos 23](#_Toc482839734)

[CU-13. Ver historial de Ingresos 24](#_Toc482839735)

[CU-14. Ver Notificaciones 25](#_Toc482839736)

[CU-15. Crear Promoción 26](#_Toc482839737)

[CU-16. Modificar Promoción 27](#_Toc482839738)

[CU-17. Eliminar Promoción 28](#_Toc482839739)

[CU-18. Registrar Asistencia 29](#_Toc482839740)

[CU-19. Cambiar Alumno de Grupo 30](#_Toc482839741)

[Modelo Relacional 31](#_Toc482839742)

[Plan de Pruebas 32](#_Toc482839743)

[Clase Alumno 32](#_Toc482839744)

[Clase Maestro 32](#_Toc482839745)

[Clase Grupo 35](#_Toc482839746)

[Clase Foto 36](#_Toc482839747)

[Clase Horario 37](#_Toc482839748)

[Clase Asistencia 39](#_Toc482839749)

[Clase Danza 39](#_Toc482839750)

[Clase Clase 42](#_Toc482839751)

[Clase Egreso 43](#_Toc482839752)

[Clase Promoción 44](#_Toc482839753)

Avance general del Proyecto de Ared Espacio

**Primera entrega 40%**

* CU-01. Registrar Alumno
* CU-02. Modificar Alumno
* CU-03. Eliminar Alumno
* CU-04. Registra Maestro
* CU-05. Modificar Maestro
* CU-06. Eliminar Maestro
* CU-07. Crear Grupo
* CU-08. Modificar Grupo
* CU-09. Eliminar Grupo

**Segunda entrega 90%**

* CU-10. Pagar Maestro
* CU-11. Registrar Egreso
* CU-12. Ver historial de Egresos
* CU-13. Ver historial de Ingresos
* CU-14. Ver Notificaciones
* CU-15. Crear Promoción
* CU-16. Modificar Promoción
* CU-17. Eliminar Promoción
* CU-18. Registrar Asistencia
* CU-19. Cambiar Alumno de Grupo

**Entrega final 100%**

* CU-20. Pagar Inscripción
* CU-21. Pagar Mensualidad

# Descripción de los casos de uso Implementados

|  |  |
| --- | --- |
| **ID:** | CU-10 |
| **Nombre:** | Pagar Maestro |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La directora registrará el pago de un maestro |
| **Precondiciones:** | PRE-1 Existe maestro en el sistema  PRE-2 Maestro tiene asignados grupo(s) |
| **Flujo Normal:** | 1. Empieza cuando la directora visita la pestaña de “Maestro” y selecciona “Pagar a Maestro”  2. Sistema despliega campo de búsqueda de maestro  3. Directora ingresa dato de búsqueda del maestro y da clic en buscar  4. Sistema despliega coincidencias  5. Directora selecciona maestro y da clic en pagar  6. Sistema calcula el monto de pago y muestra sugerencia de pago  7. Directora da clic en pagar  8. Sistema registra el pago en el sistema  9. Directora logra realizar el pago |
| **Flujo Alterno:** | FA 1 – No se encuentra maestro  4.1 Sistema despliega mensaje “No existe coincidencias”  4.2 Directora continua en el Flujo normal 3  FA 2 – Cambiar monto a pagar  7.1 Directora edita el campo de pago y da clic en pagar  7.2 Sistema registra el pago en el sistema  7.3 Directora logra realizar el pago |
| **Excepción** | EX 1 – No existe conexión con la Base de Datos  7.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde”  7.2 Directora continua en el Flujo normal 3 |
| **Postcondiciones** | POST-1 Se registra pago en la base de datos |

|  |  |
| --- | --- |
| **ID:** | CU-11 |
| **Nombre:** | Registrar Egreso |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea registrar un egreso en el sistema |
| **Precondiciones:** | Ninguno |
| **Flujo Normal:** | 1. Empieza cuando la Directora visita la pestaña de “Egresos e Ingresos” y selecciona “Registrar egreso”  2.Sistema despliega campos para ingresar monto y motivo  3. Directora ingresa los datos y da clic en “Registrar”  4. Sistema registra el egreso y muestra mensaje de “éxito”  5. Directora logra realizar un registro de egreso |
| **Flujo Alterno:** | FA 1 - No se pudo registrar por campos inválidos  4.1 Sistema no puede guardar el registro y muestra mensaje de “Error”  4.2 Usuario acepta el mensaje  4.3 Sistema continua en el Flujo normal 3 |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde”  4.2 Directora continua en el Flujo normal 3 |
| **Postcondiciones** | POST-1 Se crea registro de egreso |

|  |  |
| --- | --- |
| **ID:** | CU-12 |
| **Nombre:** | Ver historial de egresos |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea ver todos los egresos |
| **Precondiciones:** | Ninguno |
| **Flujo Normal:** | 1. Empieza cuando Directora visita la pestaña de “Egresos e Ingresos” y da clic en “Historial de Egresos”  2.Sistema despliega historial de Egresos en forma de tabla y el total acumulado  3. Directora logra ver historial de egresos |
| **Flujo Alterno:** | FA-1 Ver historial de Egresos de un mes en especifico  3.1 Directora selecciona un mes  3.2 Sistema muestra los egresos de ese mes y el total acumulado  3.3 Directora logra ver historial de egresos  FA-2 Ver historial de egresos de un año en especifico  3.1 Directora selecciona un año  3.2 Sistema muestra los egresos de ese año y el total acumulado  3.3 Directora logra ver historial de egresos |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | Ninguna |

|  |  |
| --- | --- |
| **ID:** | CU-13 |
| **Nombre:** | Ver historial de ingresos |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea ver todos los ingresos |
| **Precondiciones:** | Ninguno |
| **Flujo Normal:** | 1. Empieza cuando Directora visita la pestaña de “Egresos e Ingresos” y da clic en “Historial de Ingresos”  2.Sistema despliega historial de Ingresos en forma de tabla y el total acumulado  3. Directora logra ver historial de Ingresos |
| **Flujo Alterno:** | FA-1 Ver historial de Ingresos de un mes en especifico  3.1 Directora selecciona un mes  3.2 Sistema muestra los ingresos de ese mes y el total acumulado  3.3 Directora logra ver historial de ingresos  FA-2 Ver historial de ingresos de un año en especifico  3.1 Directora selecciona un año  3.2 Sistema muestra los ingresos de ese año y el total acumulado  3.3 Directora logra ver historial de ingresos |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | Ninguna |

|  |  |
| --- | --- |
| **ID:** | CU-14 |
| **Nombre:** | Ver Notificaciones |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea ver notificaciones de pago de alumnos o avisos de próximas clases |
| **Precondiciones:** | PRE-1 Existen Alumnos  PRE-2 Existen Grupos |
| **Flujo Normal:** | 1. Empieza cuando la directora visita la pestaña de “Otros y da clic en ”notificaciones  2. Sistema muestra notificaciones de pago y clases  3. Directora logra ver las notificaciones |
| **Flujo Alterno:** | Ninguno |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | Ninguna |

|  |  |
| --- | --- |
| **ID:** | CU-15 |
| **Nombre:** | Crear Promoción |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea crear una promoción |
| **Precondiciones:** | Ninguna |
| **Flujo Normal:** | 1. Empieza cuando la Directora visita la pestaña de “Otros” y da clic en “Promociones”  2. Sistema despliega campos de tipo, porcentaje de descuento y descripción  3. Directora da clic en nuevo y llena los campos y da clic en “Registrar”  4. Sistema guarda la promoción en el sistema y notifica éxito al registrar  5. Directora logra crear una promoción |
| **Flujo Alterno:** | EX 1 -Descripción vacía  4.1 Sistema no puede guardar la promoción, sistema notifica  4.2 Directora acepta la notificación  4.3 Sistema continua en el paso 3 del flujo normal |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | POST-1 Se crea una promoción en el sistema |

|  |  |
| --- | --- |
| **ID:** | CU-16 |
| **Nombre:** | Modificar Promoción |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea modificar una promoción |
| **Precondiciones:** | PRE-1 Existan promociones |
| **Flujo Normal:** | 1. Empieza cuando la Directora visita la pestaña de “Otros” y da clic en “Promociones”  2. Sistema despliega las promociones existentes  3. Directora selecciona una promoción  4. Sistema despliega la información de la promoción en los campos editables.  5. Directora llena los campos y da clic en “Actualizar”  6. Sistema guarda los cambios de la promoción en el sistema y notifica éxito al actualizar  7. Directora logra actualizar una promoción |
| **Flujo Alterno:** | EX 1 -Descripción vacía  6.1 Sistema no puede guardar la promoción, sistema notifica  6.2 Directora acepta la notificación  6.3 Sistema continua en el paso 5 del flujo normal |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | POST-1 Se modifica una promoción en el sistema |

|  |  |
| --- | --- |
| **ID:** | CU-17 |
| **Nombre:** | Eliminar Promoción |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La Directora desea eliminar una promoción |
| **Precondiciones:** | PRE-1 Existan Promociones |
| **Flujo Normal:** | 1. Empieza cuando la Directora visita la pestaña de “Otros” y da clic en “Promociones”  2. Sistema despliega las promociones existentes  3. Directora selecciona una promoción  4. Sistema despliega la información de la promoción en los campos editables.  5. Directora da clic en “Eliminar”  6. Sistema elimina la promoción del sistema y notifica éxito al eliminar  7. Directora logra eliminar una promoción |
| **Flujo Alterno:** | Ninguno |
| **Excepción** | EX 1 – No existe conexión con la base de datos  4.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | POST-1 Se elimina una promoción del sistema |

|  |  |
| --- | --- |
| **ID:** | CU-18 |
| **Nombre:** | Registrar Asistencia |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La directora desea registrar la asistencia de los alumnos de un grupo |
| **Precondiciones:** | PRE-1 Existan Grupos |
| **Flujo Normal:** | 1.Empieza cuando la Directora visita la pestaña de “Grupo” y da clic en “Pase de Asistencia”  2. Sistema despliega los grupos  3. Directora selecciona el grupo  4. Sistema despliega alumnos de ese grupo  5. Directora marca alumnos que asistieron y da clic en guardar  6. Sistema guarda la lista de asistencia  7. Directora logra registrar asistencia |
| **Flujo Alterno:** | Ninguno |
| **Excepción** | EX 1 – No existe conexión con la base de datos  6.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | POST-1 Se registra lista de asistencia |

|  |  |
| --- | --- |
| **ID:** | CU-19 |
| **Nombre:** | Cambiar alumno de grupo |
| **Autor:** | Mauricio Juárez Capistran |
| **Fecha de Creación:** | 14/02/17 |
| **Actores:** | Directora |
| **Descripción:** | La directora desea cambiar a un alumno de grupo |
| **Precondiciones:** | PRE-1 Existan Grupo y Alumnos |
| **Flujo Normal:** | 1.Empieza cuando la Directora visita la pestaña de “Grupo” y da clic en “Cambiar Alumno”  2. Sistema despliega los grupos origen y grupos destino  3. Directora selecciona el grupo origen  4. Sistema despliega alumnos de ese grupo  5. Directora selecciona el o los alumnos a cambiar  6 Sistema los mueve a una nueva lista  7. Directora selección el grupo destino y da clic en “Cambiar Alumno(s)”  8. Sistema cambia al o los alumnos de grupo y notifica éxito al cambiar.  9. Directora logra cambiar a un alumno o alumnos |
| **Flujo Alterno:** | Ninguno |
| **Excepción** | EX 1 – No existe conexión con la base de datos  6.1 Sistema despliega mensaje “No hay conexión, inténtalo más tarde” |
| **Postcondiciones** | POST-1 Se cambia a alumno(s) de grupo |

# Diagramas de Robustez

## CU-10. Pagar Maestro



## CU-11. Registrar Egreso



## CU-12. Ver historial de Egresos



## CU-13. Ver historial de Ingresos



## CU-14. Ver Notificaciones



## CU-15. Crear Promoción



## CU-16. Modificar Promoción



## CU-17. Eliminar Promoción



## CU-18. Registrar Asistencia



## CU-19. Cambiar Alumno de Grupo



# Modelo de Dominio



# Diagrama de Clases



# Diagramas de Secuencia

## CU-10. Pagar Maestro

## CU-11. Registrar Egreso

## CU-12. Ver historial de Egresos



## CU-13. Ver historial de Ingresos



## CU-14. Ver Notificaciones

## CU-15. Crear Promoción

## CU-16. Modificar Promoción



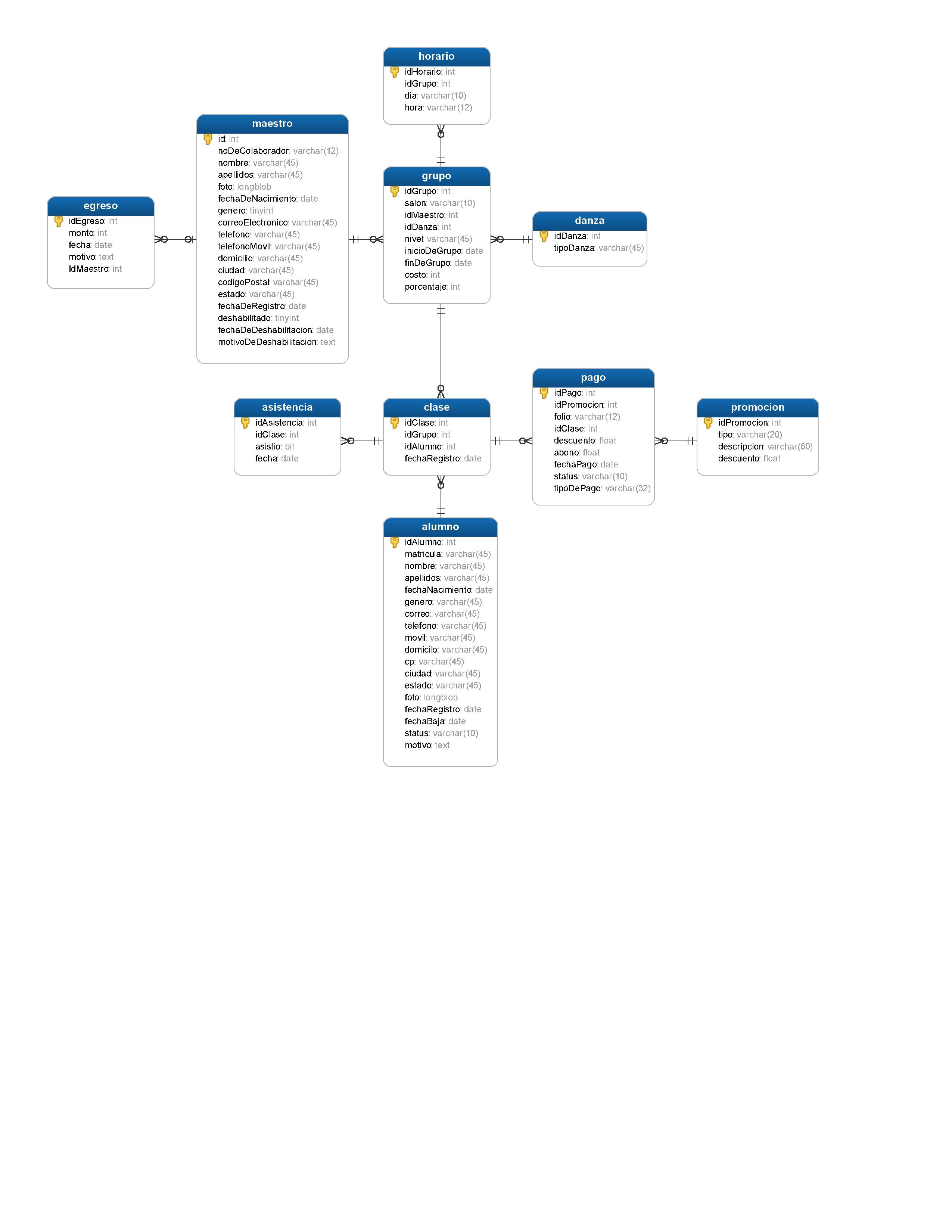
## CU-17. Eliminar Promoción

## CU-18. Registrar Asistencia

## CU-19. Cambiar Alumno de Grupo



# Modelo Relacional



# Plan de Pruebas

## Clase Alumno

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obteneCoincidenciasDeAlumnoPorApellido | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Apellido=”Sanchez” | El apellido existe en la base de datos de alumnos. | El sistema regresa la lista de estudiantes | El sistema regresa los datos de los estudiantes en las tablas. | V |
| Apellidos=  “Okumura” | El apellido no existe en la base de datos. | El sistema no regresa nada. | El sistema no regresa nada en las tablas | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerCoincidenciasPorMatricula | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| matricula=  “01”; | La matrícula existe en la base de datos de alumnos registrados. | El sistema regresa la lista de estudiantes | El sistema regresa los estudiantes en las tablas. | V |
| Matricula=  “”0123456789; | La matrícula no existe en la base de datos de alumnos. | El sistema no regresa nada. | El sistema no regresa nada en las tablas | V |

## Clase Maestro

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: ActualizarMaestro | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| obtenerMaestroPorApellido ("Sanchez") | El maestro existe en la base de datos | Los datos del maestro seleccionado son editables | El sistema manda el mensaje de que se actualizo correctamente | V |
| obtenerMaestroPorApellido("Hoyos") | El maestro no existe en la base de datos | La tabla de datos de maestro se quedan vacios. | El sistema manda mensaje de que el maestro no existe o el dato es erróneo. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: Registrar Maestro | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Nombre("Pedro");  Apellidos("Sanchez");  Ciudad("Xalapa");  CodigoPostal("91315");  CorreoElectronico("pedro\_12345@hotmail.com");  Deshabilitado(false);  Domicilio("Av. Xalapa #6");  Estado("Veracruz");  Foto(null);  Genero(true);  Telefono("1234567890");  TelefonoMovil("1234567890"); | Los campos correctos para registrar a un maestro | El maestro aparece registrado en la base de datos | Mensaje de éxito al registrar al maestro | V |
| Nombre("213123");  Apellidos("true");  Ciudad("Xalapa");  CodigoPostal("Sanchez");  CorreoElectronico("noselaneta@hotmail.com");  Deshabilitado(true);  Domicilio("Av. Xalapa #6");  Estado("Veracruz");  Foto(null);  Genero(false);  Telefono("abcdefgh");  TelefonoMovil("1234567890"); | Los datos de los campos a ingresar tienen valores incorrectos. | El sistema manda un mensaje de advertencia. | El sistema manda un mensaje de que se debe verificar los datos de los campos para poder  Registrar. | V |
| Nombre("");  Apellidos("");  Ciudad("");  CodigoPostal("");  CorreoElectronico("");  Deshabilitado(false);  Domicilio("");  Estado("");  Foto(null);  Genero(true);  Telefono("");  TelefonoMovil(""); | Los campos vacíos para registrar un maestro | Mensaje de advertencia | El sistema despliega un mensaje de advertencia, de que los campos están vacíos | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerMaestroPorNoDeColaborador | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| noDecolaborador=  2213 | El apellido existe en la base de datos de alumnos. | El sistema regresa la lista de estudiantes | El sistema regresa los estudiantes en las tablas. | V |
| noDecolaborador=  000 | El apellido existe en la base de datos de alumnos. | El sistema no regresa nada. | El sistema no regresa nada en las tablas | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerMaestroPorNombre | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| nombre=  Pedro | El nombre existe en la base de datos de alumnos. | El sistema regresa la lista de estudiantes | El sistema regresa los estudiantes en las tablas. | V |
| nombre=  pánfilo | El nombre existe en la base de datos de alumnos. | El sistema no regresa nada. | El sistema no regresa nada en las tablas | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerMaestroPorApellidos | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Apellidos=  Sanchez | El apellido existe en la base de datos de alumnos. | El sistema regresa la lista de estudiantes | El sistema regresa los estudiantes en las tablas. | V |
| Apellidos=  Hoyos | El apellido existe en la base de datos de alumnos. | El sistema no regresa nada. | El sistema no regresa nada en las tablas | V |

## Clase Grupo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: CrearGrupo | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Nivel("Basico");  Salon("CC3");  TipoDeDanza("Ballet");  Maestro maestro = Maestro.obtenerMaestroPorApellido("").get(0);  grupo.setIdMaestro(maestro); ArrayList<Horario> horarios = new ArrayList<>();  boolean resultado = grupo.crear(horarios); | El grupo no existe en la base de datos. | El grupo se verá reflejado en la tabla. | El sistema manda un mensaje de que el grupo fue creado exitosamente. | V |
| setNivel("");  setSalon("");  grupo.setTipoDeDanza("");  Maestro maestro = Maestro.obtenerMaestroPorApellido("").get(0);  grupo.setIdMaestro(maestro);  ArrayList<Horario> horarios = new ArrayList<>();  boolean resultado = grupo.crear(horarios); | El grupo no existe en la base de datos, pero los campos están vacios. | El sistema manda una advertencia. | El sistema manda una advertencia de que los campos están vacios y deben verificarse. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: ActualizarGrupo | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Nivel("Basico");  Salon("CC3");  TipoDeDanza("Ballet");  Maestro maestro = Maestro.obtenerMaestroPorApellido("").get(0);  grupo.setIdMaestro(maestro); ArrayList<Horario> horarios = new ArrayList<>();  boolean resultado = grupo.crear(horarios); | El grupo no existe en la base de datos. | El grupo se verá reflejado en la tabla. | El sistema manda un mensaje de que el grupo fue creado exitosamente. | V |
| setNivel("");  setSalon("");  grupo.setTipoDeDanza("");  Maestro maestro = Maestro.obtenerMaestroPorApellido("").get(0);  grupo.setIdMaestro(maestro);  ArrayList<Horario> horarios = new ArrayList<>();  boolean resultado = grupo.crear(horarios); | El grupo no existe en la base de datos, pero los campos están vacíos. | El sistema manda una advertencia. | El sistema manda una advertencia de que los campos están vacíos y deben verificarse. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerTodosLosGrupos | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| ArrayList<Grupo> grupos = new ArrayList<>(Grupo.listarGrupos()); | Hay grupos registrados en la base de datos | El sistema regresa la lista de grupos | EL sistema muestra los grupos registrados | V |
| ArrayList<Grupo> grupos = new ArrayList<>(Grupo.listarGrupos()); | Aun no hay grupos registrados. | El sistema manda un mensaje de advertencia. | El sistema manda un mensaje de que no hay algún grupo registrado. | V |

## Clase Foto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: obtenerImagenCorrecta | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| ruta = "C:\\Users\\Mauricio Juárez\\Desktop\\Imagen\_TEST.jpg"; ()); | La ruta de la imagen a subir está bien escrita. | El sistema obtiene la imagen correctamente de la ruta. | Se muestra la imagen elegida en la ventana. | V |
| ruta = ""; | El campo de la ruta esta vacío o no se eligió alguna | El sistema manda una advertencia. | El sistema despliega el mensaje de advertencia de que no hay alguna coincidencia con el archivo que se busco. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: AgregarImagenCorrecta | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| ruta = "C:\\Users\\Mauricio Juárez\\Desktop\\Imagen\_TEST.jpg"; | La ruta de la imagen está en el formato correcto. | El sistema obtiene la ruta correcta | Se muestra la imagen elegida en la ventana. | V |
| ruta = "file:C:Imagen\_TEST.jpg"; | La ruta de la imagen está en un formato incorrecto. | El sistema manda una advertencia. | El sistema despliega el mensaje de advertencia y la venta se queda vacía, donde se mostraría la foto elegida. | V |

## Clase Horario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: CrearHorario | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| horario.setDia("Lunes"); horario.setHora("12:00-14:00");  Grupo grupo = Grupo.listarGrupos().get(0);  horario.setIdGrupo(grupo);  boolean resultado = horario.crear(); | Los datos de horario de un grupo no han sido registrados en la base de datos. | El sistema muestra el horario agregado al grupo. | El sistema manda un mensaje de que el registro del grupo fue exitoso. | V |
| Horario horario = new Horario();  horario.crear(); | Los datos de horario están en blanco. | El sistema manda una advertencia. | El sistema despliega el mensaje de advertencia de que se debe ingresar datos correctos en los campos solicitados. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: ActualizarHorario | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Horario horario = grupo.getHorarioList().get(0);  horario.setDia("Martes");  boolean resultado = horario.actualizar(); | El horario del grupo solicitado está registrado en la base de datos. | El sistema muestra los datos del horario y los vuelve editables. | El sistema manda un mensaje de que el horario se actualizo exitosamente. | V |
| Horario horario = new Horario();  horario.actualizar(); | El horario del grupo solicitado es incorrecto o no existe. | Las tablas de horarios se quedan vacias. | El sistema despliega el mensaje de advertencia de que el horario del grupo solictado, es incorrecto o no existe. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función: EliminarHorario | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Grupo grupo = Grupo.listarGrupos().get(0);  Horario horario = grupo.getHorarioList().get(0);  boolean resultado = horario.eliminar(); | El horario solicitado existe en la base de datos. | El sistema elimina el horario seleccionado de la base de dstos. | El sistema muestra un mensaje de horario eliminado satisfactoriamente. | V |
| Horario horario = new Horario();  horario.eliminar(); | El horario solicitado no existe en la base de datos | Las tablas se quedan vacías. | El sistema despliega el mensaje de advertencia donde indica que el horario solicitado no existe. | V |

## Clase Asistencia

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:Crear Asistencia | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Asistencia asistencia = new Asistencia();  asistencia.setAsistio(true);  asistencia.setFecha(new Date(new GregorianCalendar().getTimeInMillis()));  asistencia.setIdClase(Clase.listaDeClases().get(0));  boolean resultado = asistencia.crear();  Assert.assertTrue("Prueba registrar asistencia correctamente", resultado);  } | Se selecciona un grupo y el pase de asistencia esta limpio o libre. | El sistema registra la fecha y marca que un alumno asistió ese dia, | El sistema marca la casilla del nombre seleccionado con un circulo negro y manda y un mensaje de registro de asistencia exitoso. | V |
| Asistencia asistencia = new Asistencia();  boolean resultado = asistencia.crear();  Assert.assertTrue("Prueba registrar asistencia incorrecta", resultado); | El pase de asistencia esta libre y no se toma la lista de alumnos de una clase. | No se toma la fecha, ni la lista de alumnos de una clase. | El sistema manda un mensaje de que no hay conexión a la base de datos y no muestra nada. | V |

## Clase Danza

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ObtenerTodaslasDanzas | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| List<Danza> lista = Danza.obtenerTodas();  boolean resultado = !lista.isEmpty();  assertTrue("Prueba obtener todas las danzas", resultado); | Se manda la cadena vacia para buscar todas las danzas. | El sistema regresa todas los tipos de danzas registradas en la bd. | El sistema muestra en la tabla de danzas, todos los tipos de danza registrados. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:CrearDanza | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Danza danza = new Danza();  danza.setTipoDanza("Prueba Crear danza");  boolean resultado = danza.crear();  assertTrue("Prueba crear danza correctamente", resultado);  } | Se mandan los datos para crear la danza correctamente | El sistema registra la danza en la base de datos | El sistema muestra la danza registrada en la bd y manda mensaje de que fue registrada correctamente | V |
| Danza danza = new Danza();  boolean resultado = danza.crear();  assertTrue("Prueba crear danza incorrectamente", resultado); | Se manda los datos cuando no hay conexión o se manda incorrectamente. | El sistema no registra nada. | El sistema manda un mensaje de que hubo un error de conexión. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ActualizarDanza | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Danza danza = Danza.obtenerTodas().get(0);  danza.setTipoDanza("Prueba actualizar danza");  boolean resultado = danza.actualizar();  assertTrue("Prueba actualizar danza conrrectamente", resultado); | Se manda a llamar la función con los parámetros vacíos | El sistema obtiene todos los tipos de danza en la base de datos | El sistema muestra las danzas registradas en la bd y habilita campos para editar. | V |
| Danza danza = new Danza();  boolean resultado = danza.actualizar();  assertTrue("Prueba actualizar danza inconrrectamente", resultado); | Se manda los datos para la función cuando no hay conexión o se manda incorrectamente. | El sistema no obtiene nada de la base de datos. | El sistema manda un mensaje de que no se pudo conectar a la base de datos o no existe el tipo de danza. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:EliminarDanza | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Danza danza = Danza.obtenerTodas().get(0);  boolean resultado = danza.eliminar();  assertTrue("Prueba eliminar danza conrrectamente", resultado); | Se selecciona una danza y se manda a llamar a la función eliminar | El sistema borra el tipo de danza de la base de datos | El sistema manda un mensaje de que el tipo de danza fue eliminado correctamente | V |
| Danza danza = new Danza();  Danza danza = new Danza();  boolean resultado = danza.eliminar();  assertTrue("Prueba eliminarr danza inconrrectamente", resultado); | Se manda los datos para la función cuando no hay conexión o se manda incorrectamente. | El sistema no puede acceder a la base de datos. | El sistema manda un mensaje de que no se pudo conectar a la base de datos o no existe el tipo de danza. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:BuscarDanzaporNombre | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Danza danza = Danza.obtenerTodas().get(0);  Danza danzaEncontrada = Danza.buscarPorTipoDanza(danza.getTipoDanza());  Assert.assertEquals("Prueba buscar danza por nombre de la danza",danzaEncontrada, danza); | Se manda la cadena con un nombre de tipo de danza para buscar en la base de datos | El sistema regresa el tipos de danza registradas en la bd, con ese nombre si la hay. | El sistema muestra en la tabla de danzas, todos los datos del tipo de danza seleccionado | V |

## Clase Clase

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ObtenerClasesPorGrupo | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| List<Clase> clases = Clase.obtenerClasesDelGrupo(17);  boolean resultado = clases.size() > 0;  Assert.assertTrue("Prueba obtener Clases por Id de grupo", resultado); | Se manda la id de la clase de danza para buscar en la base de datos | El sistema regresa la clase registrada en la bd, con ese nombre si la hay. | El sistema muestra en la tabla de clases, todos los datos del tipo de danza seleccionado | V |
| List<Clase> clases = Clase.obtenerClasesDelGrupo(-1);  boolean resultado = clases.isEmpty();  Assert.assertTrue("Prueba No encontrar Clases por Id de grupo", resultado); | Se manda un id invalido de una clase de danza | El sistema no puede obtener los datos. | El sistema regresa un mensaje de advertencia de que la clase no existe | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:CambiarDeGrupo | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Clase clase = Clase.obtenerClasesDelGrupo(6).get(0);  Grupo grupo = Grupo.listarGrupos().get(0);  boolean resultado = clase.cambiarDeGrupo(grupo);  Assert.assertTrue("Prueba cambiar grupo en clase", resultado); | Se manda la id de la clase de danza origen, además del ide del alumno seleccionado y el grupo a donde se quiere cambiar el alumno | El sistema ejecuta la transacción reduciendo el numero de alumnos en el origen y copiando y aumentando el numero de alumnos en el destino | El sistema muestra en la tabla de clases, todos los datos de los alumnos  así como el recién transferido. | V |
| Clase clase = Clase.obtenerClasesDelGrupo(6).get(0);  boolean resultado = clase.cambiarDeGrupo(new Grupo());  Assert.assertTrue("Prueba cambiar grupo en clase incorrectamente", resultado); | Se manda un id de una clase que no existe o no se selecciona a un alumno | El sistema no puede acceder a la base de datos, dado que no existe la clase destino | El sistema manda un mensaje de que la clase destino no existe o que el alumno ya fue transferido | V |

## Clase Egreso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:CrearEgreso | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Egreso egreso = new Egreso();  egreso.setMonto(0);  egreso.setMotivo("Prueba ");  egreso.setFecha(new Date(new GregorianCalendar().getTimeInMillis()));  boolean resultado = egreso.crear();  assertTrue("Prueba registrar egreso correctamente", resultado); | Se ingresa un monto, un motivo y se elige la opción de crear egreso. | El sistema registra el egreso en la base de datos, | El egreso es visible en la base de datos. | V |
| Egreso egreso = new Egreso();  boolean resultado = egreso.crear();  assertTrue("Prueba registrar egreso incorrectamente", resultado); | Se selecciona la opción de crear un egreso sin mandar algún dato. | El sistema no puede registrar los datos. | El sistema regresa un mensaje de advertencia de que se tienen que llenar los campos. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ObtenerTodosLosEgresos | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| List<Egreso> lista = Egreso.obtenerTodosLosEgresos();  boolean resultado = !lista.isEmpty();  assertTrue("Prueba obtener todos los egresos",resultado); | Se manda a llamar a la función que obtiene todos los egresos registrados en la base de datos, no es necesario algún parámetro. | El sistema regresa todos los egresos registrados en la bd. | El sistema muestra en la tabla de egresos, todos los datos del egreso | V |

## Clase Promoción

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:CrearPromocion | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Promocion promocion = new Promocion();  promocion.setDescripcion("Promocion prueba");  promocion.setDescuento(0.4f);  promocion.setTipo("Mensualidad");  boolean resultado = promocion.crear();  assertTrue("Prueba crear promocion correctamente", resultado); | Se ingresa una descripción, un descuento y se elige la mensualidad, después se elige crear promoción. | El sistema registra la promoción en la base de datos, | La promoción es visible y usable en la base de datos. | V |
| Promocion promocion = new Promocion();  boolean resultado = promocion.crear();  assertTrue("Prueba crear promocion correctamente", resultado); | Se selecciona la opción de crear una promoción sin mandar algún dato. | El sistema no puede registrar los datos. | El sistema regresa un mensaje de advertencia de que se tienen que llenar los campos. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ActualizarPromocion | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Promocion promocion = Promocion.listaDePromociones().get(0);  promocion.setDescripcion("Prueba actualizar promocion");  promocion.setDescuento(0.15f);  promocion.setTipo("Inscripción");  boolean resultado = promocion.actualizar();  assertTrue("Prueba actualizar promocion correctamente", resultado); | Se ingresa una descripción, un descuento y se elige el tipo de la promoción después se elige actualizar promoción. | El sistema actualiza  la prómocion en la base de datos, | La promocion es visible en la base de datos. | V |
| Promocion promocion = new Promocion();  boolean resultado = promocion.actualizar();  assertTrue("Prueba actualizar promocion incorrectamente", resultado); | Se selecciona la opción de crear una promoción sin mandar algún dato. | El sistema no puede actualizar los datos. | El sistema regresa un mensaje de advertencia de que se tienen que llenar los campos. | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:EliminarPromocion | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| Promocion promocion = Promocion.listaDePromociones().get(0);  boolean resultado = promocion.eliminar();  assertTrue("Prueba eliminar promocion correctamente", resultado); | Se selecciona una promoción y se elige eliminar | El sistema borra la promoción de la base de datos | La promoción desaparece de las disponibles | V |
| Promocion promocion = new Promocion();  boolean resultado = promocion.eliminar();  assertTrue("Prueba eliminar promocion correctamente", resultado); | Se selecciona la opción de eliminar sin elegir un dato. | El sistema no puede acceder a los datos. | El sistema regresa un mensaje de advertencia de que se tienen seleccionar una promoción | V |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función:ObtenerTodasLasPromociones | | | | |
| Entradas | Condiciones de Entrada | Salida | Salida Esperada | Resultado |
| List<Promocion> lista = Promocion.listaDePromociones();  boolean resultado = !lista.isEmpty();  assertTrue("Prueba obtener todos los resultados", resultado); | Se llama a la función y se obtienen los datos de promociones registradas | El sistema obtiene los datos de todas las promociones disponibles | Las promociones son visibles en la tabla de promociones | V |