## Pagos de profesor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | obtenerPagos(int idProfesor): List<PagoProfesor> | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| idProfesor: 1. | Ninguna excepción relativa a la persistencia. | No hay excepciones. | * El profesor con el id ‘1’ está registrado. * Existen o no pagos del profesor registrados. | Lista válida. |
| idProfesor: -1. | NullPointer-  Exception | NullPointer-  Exception | * El profesor con el id ‘-1’ no existe. | Lista vacía. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | registrarPago(PagoProfesor pago, int idProfesor): boolean | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| PagoProfesor:   * tipoPago: true. * monto: 900.5.   idProfesor: 1. | True. | N/A. | * PagoProfesor con todas sus propiedades establecidas correctamente. * El profesor con el id ‘1’ está registrado. | True. |
| PagoProfesor:   * tipoPago: true. * monto: 3t.   idProfesor: 1. | False. | N/A. | * PagoProfesor con un monto no válido. * El profesor con el id ´1´ está registrado. | False. |
| PagoProfesor:   * tipoPago: true. * monto: 900.5.   idProfesor: -1. | False. | N/A. | * El profesor con el id ‘-1’ no existe. | False. |

Edición promociones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | editarPromocion(Promocion promocion): boolean | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| Promocion:   * Descripción: promoción de prueba. * idPromocion: 1 * idProfesor: 1 * Nombre: promoción de decuento editada. * Porcentaje: 400 | True. | N/A. | Se espera la adicción de una promoción existente. | True. |
| Promocion:   * Descripción: promoción de prueba. * idPromocion: 1 * idProfesor: 0 * Nombre: promoción de decuento editada.   Porcentaje: 20 | False. | N/A. | Se espera que no se cree un registro de pago con el id de un profesor inexistente. | False. |

Registro de promociones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | registrarPromocion(Promocion promocion): boolean | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| * Nombre: promoción de prueba * idPromocion: 0 * idProfesor:   1   * Descripción: promoción de descuento. * Porcentaje: 20 | True. | N/A. | Se espera el registro correcto de una promoción con un id de profesor existente. | True. |
| * Nombre: promoción de prueba * idPromocion: 0 * idProfesor:   - 1   * Descripción: promoción de descuento.   Porcentaje: 20 | False. | N/A. | Se espera que no se cree un registro de una promoción con un id de un profesor no existente. | False. |

Obtener promociones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | ObtenerPromociones(int idProfesor): list<promocion> | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| * Id profesor: 1 | List <Promocion> | N/A. | Se espera una lista de promociones registradas por un profesor. | List <Promocion> |
| * Id profesor: 0 | NullPointer Exception. | N/A. | Se espera que se lance una excepción sobre un id de profesor que no existe. | NullPointer Exception. |

Registrar pago de alumno.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | registrarPago(PagoAlumno pago, idProfesor, int idAlumno, int idPromocion: boolean | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| * Pago alumno: fecha: new Date(). idPago: 0. Monto: “500”   idTipoPago: 1   * idAlumno: 1 * idPromocion: 1 * idProfesor: 1 | True. | N/A. | Se espera un registro de un nuevo pago de alumno con todos los atributos pertenecientes registrados por un profesor con el id indicado. | True. |
| * Pago alumno: fecha: new Date(). idPago: 0. Monto: “no es un pago”   idTipoPago: 1   * idAlumno: 1 * idPromocion: 1 * idProfesor: 1 | False. | N/A. | Se espera que no se registre un pago de alumno con una cantidad no valida o con incoherencia de datos. | False. |
| * Pago alumno: fecha: new Date(). idPago: 0. Monto: “500”   idTipoPago: 1   * idAlumno: 1 * TidPromocion: 0 * idProfesor: 1 | True. | N/A. | Se espera el registro de un pago de alumno sin la propiedad de promoción con un id de profesor indicado. | True. |

Consultar pagos de alumno.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | obtenerPagos(int idAlumno, int idProfesor) List<PagoAlumno> | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| * idAlumno. 1 * idProfesor: 1 | List <PagoAlumno> | N/A. | Se espera una lista de pagos de un alumno y exista un id con el profesor indicado. | List <PagoAlumno> |
| * idAlumno: 0 * idProfesor: 1 | NullPointer Exception. | N/A. | Se espera que el proceso arroje una excepción de tipo null debido a la no existencia del alumno. | NullPointer Exception. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | obtenerPagos(int idProfesor) List<PagoAlumno> | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| idProfesor: 1 | List <PagoAlumno> | N/A. | Se espera una lista de pagos de un alumno con el profesor indicado. | List <PagoAlumno> |

Obtener promoción.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | obtenerPromocion(int idPromocion): Promocion | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| * idPromocion. 1 | Promocion. | N/A. | Para que esta prueba funcione debe existir en base de datos una promoción con ese id. | Promocion |
| * idPromocion: 0 | NullPointer Exception. | N/A. | Para que esta prueba funcione no debe existir en base de datos una promoción con ese id. | NullPointer Exception. |

Obtener rentas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | obtenerRentas(int idCliente): List<Renta> | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| idUsuario: 1 | List<Renta> | N/A | Deben existir registros previos de rentas de ese cliente. | List<Renta> |
| idUsuario: 2 | NullPointer Exception | N/A | El cliente ingresado no debe tener registros de rentas. | NullPointer  Exception |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | registrarRenta(Renta renta): boolen | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| Renta:   * fecha: 2018-05-12. * idCliente: 1. * idRenta: 0. * Monto: 250. * Dia:   + Dia: Lunes.   + horaFin: 12:00.   + horaInicio:   + 11:30.   + Id: 0.   + Tipo: false.   + Salon: X. | True. | True. | * El cliente con id 1 existe en base de datos. * Todos los atributos de la renta están establecidos correctamente. * Las horas de inicio y fin del día no chocan con otro horario. | True |
| Renta:   * fecha: 2018-05-12. * idCliente: 1. * idRenta: 0. * Monto: w. * Dia:   + Dia: Lunes.   + horaFin: 12:00.   + horaInicio:   + 11:30.   + Id: 0.   + Tipo: false.   + Salon: X. | False. | False. | * El cliente con id 1 existe en base de datos. * El monto no es válido. * Las horas de inicio y fin del día no chocan con otro horario. | False. |
| Renta:   * fecha: 2018-05-12. * idCliente: 1. * idRenta: 0. * Monto: 250. * Dia:   + Dia: Lunes.   + horaFin: 18:00.   + horaInicio: 17:30.   + Id: 0.   + Tipo: false.   + Salon: X. | Horario Exception. | “Las horas establecidas chocan con otro horario”. | * El cliente con id 1 existe en base de datos. * Todos los atributos de la renta están establecidos correctamente. * Las horas de inicio y fin del día chocan con otro horario. | Horario Exception. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | editarRenta(Renta renta): boolen | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| Renta:   * Monto: 250. * Dia:   + horaFin: 12:00.   + horaInicio:   + 11:30. | True. | True. | * Existe al menos un cliente registrado. * Existe al menos una renta registrada para el cliente. * Todos los atributos de la renta están establecidos correctamente. * Las horas de inicio y fin del día no chocan con otro horario. | True |
| Renta:   * Id: 0. * Monto: 250. * Dia:   + horaFin: 12:00.   + horaInicio: 11:30. | False. | False. | * La renta con el id 0 no existe. | False. |
| Renta:   * Monto: w. * Dia:   + horaFin: 12:00.   + horaInicio:   + 11:30. | False. | False. | * El monto no es válido. | False. |
| Renta:   * Monto: 250. * Dia:   + horaFin: 12:00.   + horaInicio: 10:00. | Horario Exception. | “Las horas establecidas chocan con otro horario”. | * Las horas establecidas chocan con otro horario. | Horario Exception. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | List<Renta> obtenerRentas(). | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| N/A. | Ninguna excepción relativa a la persistencia. | Ninguna excepción relativa a la persistencia. | Existen o no rentas registradas. | Lista válida. |

## Pagos temporales.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | List<PagoTemporal> obtenerPagos(). | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| N/A. | Ninguna excepción relativa a la persistencia. | Ninguna excepción relativa a la persistencia. | Existen o no pagos registrados. | Lista válida. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | boolean eliminarPago(int idPago). | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| - idPago: id del primer pago en los registros. | True. | True. | Existe al menos un pago registrado. | True. |
| - idPago: 0. | False. | False. | El pago con el id 0 no existe. | False. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Función** | boolean registrarPago(PagoTemporal pago). | | | |
| **Entradas** | **Salidas esperadas** | **Salidas obtenidas** | **Condiciones de salida** | **Resultado** |
| Pago:   * idPago: 0. * tipoPago: 1. * Fecha: 2018-05-24. * Monto: 2500. * Profesor: primer profesor en los registros. * Alumno: primer alumno registrado en el primer grupo registrado del profesor. | True. | True. | - Existe al menos un profesor registrado.  - Existe al menos un grupo registrado para el profesor.  - Existe al menos un alumno inscrito al grupo del profesor.  - Los atributos del pago son válidos. | True. |
| Pago:   * idPago: 0. * tipoPago: 1. * Fecha: 2018-05-24. * Monto: “”. * Profesor: primer profesor en los registros. * Alumno: primer alumno registrado en el primer grupo registrado del profesor. | False. | False. | El valor del monto no es válido. | False. |
| Pago:   * idPago: 0. * tipoPago: 1. * Fecha: 2018-05-24. * Monto: 2500. * Profesor: primer profesor en los registros. * Alumno:   + Id: 0. | False. | False. | El alumno con el id 0 no existe. | False. |
| Pago:   * idPago: 0. * tipoPago: 1. * Fecha: 2018-05-24. * Monto: 2500. * Profesor:   + Id: 0. * Alumno: primer alumno registrado en el primer grupo registrado del profesor. | False. | False. | El profesor con el id 0 no existe. | False. |