Proyecto - Desarrollo de software “FashionProject” para la empresa “Ximena&Carolina”

Universidad Austral de Chile
Facultad de Ciencias de la Ingeniería.
Instituto de Ingeniería Civil Informática

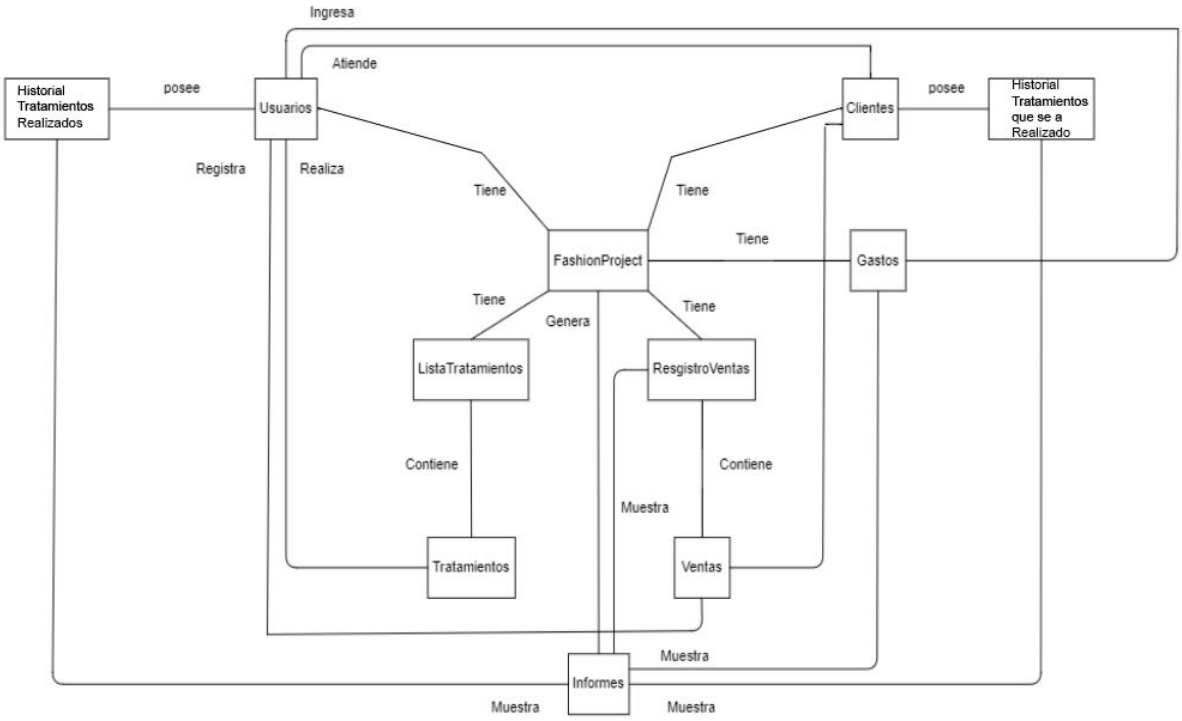
Desarrollado por:
- **-Camilo Alarcón**
- **-Humberto Campos**
- **-Gabriel Galilea**
- **-María José Núñez**

Profesores colaborantes:
- **-Valeria Henriquez**
- **-Raimundo Vega**

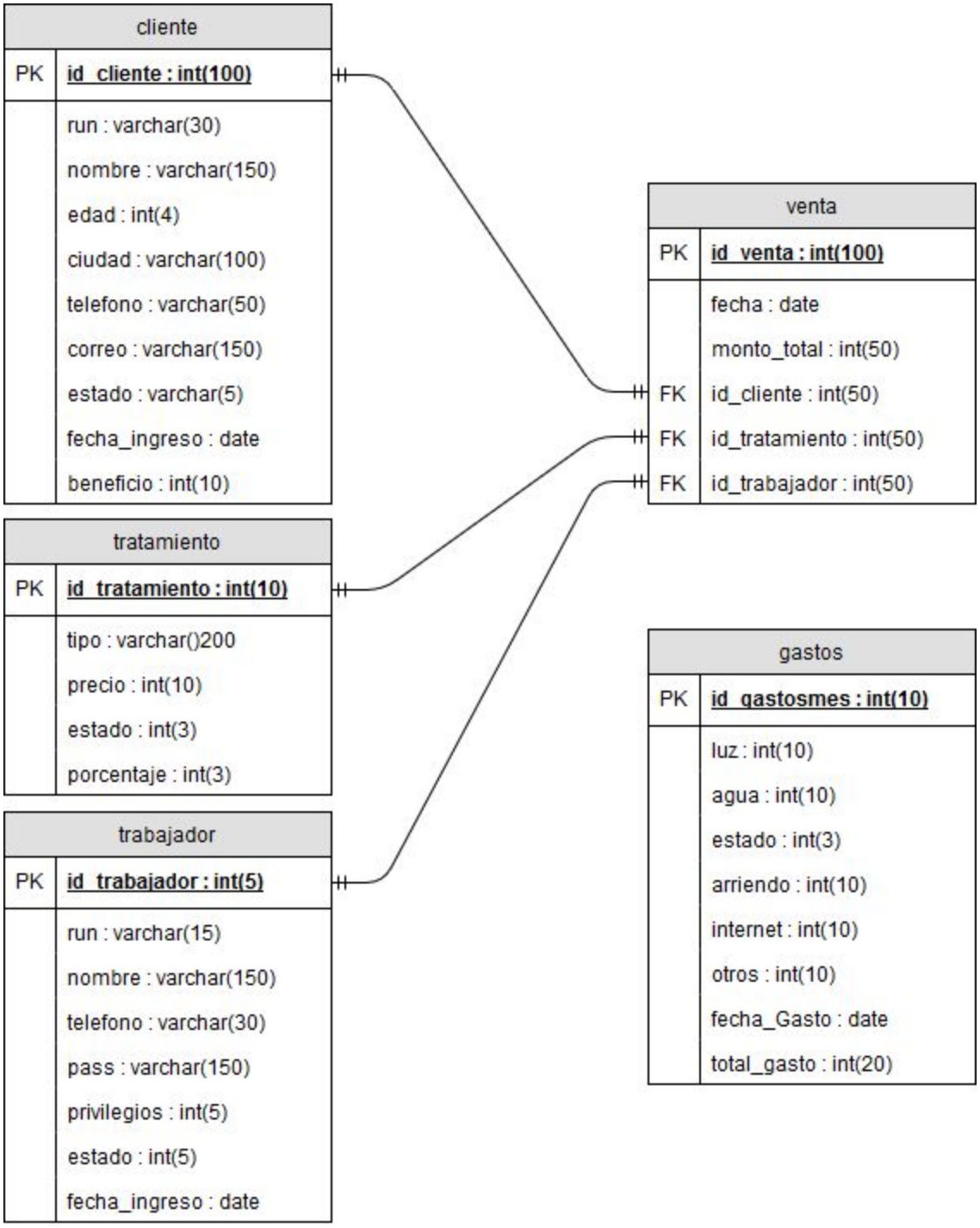
Fecha:
- **-18 de diciembre del 2017**

Indice

Modelo Conceptual FashionProject



Base de datos



Incremento 1:

Caso de uso: Gestión Clientes.

Actores: Usuario administrador, usuario común.
Precondición: El usuario debe estar registrado en el sistema.

OBS: para los cursos alternativos de modificar y "eliminar" el cliente debe estar registrado en la base de datos.

Resumen: El usuario ingresa al sistema y gestiona a los clientes, es decir, puede ingresar un nuevo cliente o también puede modificarlos y "eliminarlos".
Tipo: Primario - esencial

Curso normal de los eventos: Ingresar cliente

Actor	Sistema
1) El caso de uso comienza cuando el usuario finaliza el tratamiento del cliente.	
2) El usuario accede al sistema.	3) El sistema autoriza el ingreso del usuario.
	4) Sistema despliega el menú.
5) Usuario ingresa a la gestión del cliente.	6) Sistema despliega la ventana de gestión de clientes.
7) Usuario solicita los datos al cliente y los ingresa.	8) Sistema verifica los datos ingresados.
	9) El cliente es registrado, desplegando un mensaje.
	10) Muestra la lista actualizada de los clientes.
11) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Cliente ya existente en el sistema.

Actor	Sistema
	9*) El sistema despliega un mensaje diciendo que el cliente ya se encuentra registrado.
10) Usuario corrige los datos o cancela la operación.	

Curso alternativo: Modificar cliente

Actor	Sistema

7*) Busca un cliente al cual modificar sus datos.	8) Muestra al cliente que coincide con la búsqueda.
9) Selecciona el cliente.	10) Despliega los datos almacenados del cliente.
11) Usuario solicita datos al cliente y los ingresa.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de clientes.
14) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Curso alternativo de “Modificar cliente”.

Actor	Sistema
11**) Usuario solicita datos al cliente y los ingresa de forma incompleta.	12) Despliega un mensaje de error.
13) Usuario completa todos los campos y los ingresa.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de clientes.
14) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Eliminar cliente.

Actor	Sistema
7*) Usuario busca un cliente para eliminarlo.	8) Muestra al cliente que coincide con la búsqueda.
9) Selecciona el cliente.	10) Despliega los datos almacenados del cliente.
11) Selecciona la opción de eliminar.	12) Despliega mensaje de confirmación.
13) Confirma su selección.	14) Cambia el estado del cliente y no lo despliega en la lista de clientes.
	15) Se muestra la lista actualizada de los clientes.
16) El usuario realiza otra acción o sale del sistema.	

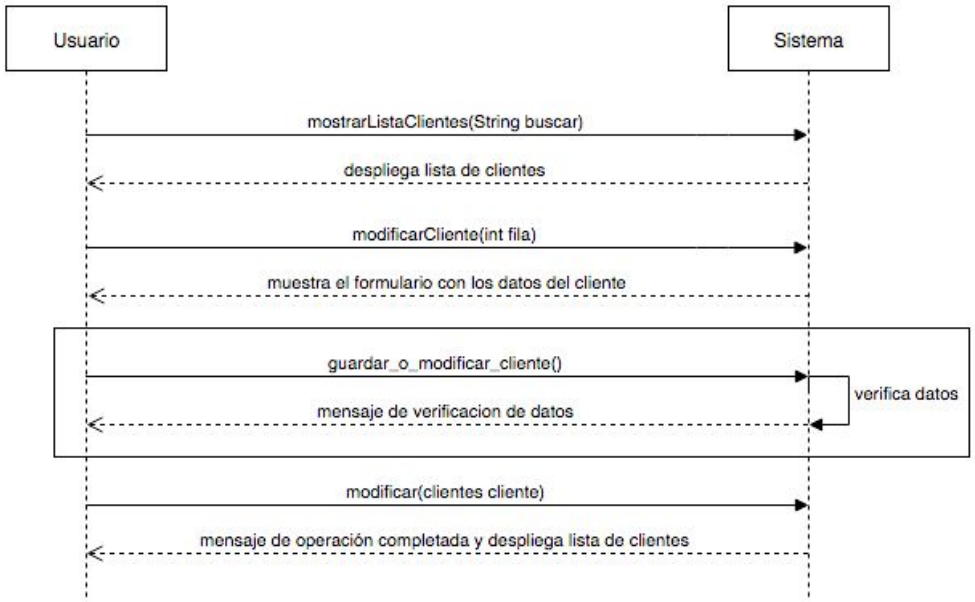
Curso alternativo: Curso alternativo de “Eliminar Cliente”.

Actor	Sistema
13**) Usuario se percata que seleccionó otro cliente y cancela la operación.	

Diagramas de Secuencia:

Ingresar Cliente

Modificar Cliente



Eliminar Cliente

Contratos:

- Ingresar cliente

Nombre: mostrarListaClientes(String buscar)
Responsabilidades: Mostrar lista con los clientes registrados
Precondiciones:
Post-condiciones: se busca la lista de los objetos cliente

Nombre: nuevoCliente()
Responsabilidades: Limpiar el formulario para luego habilitarlo para ingresar los datos.
Precondiciones:
Post-condiciones: Habilitar los atributos para almacenar datos del cliente (setea los campos “” y los activa).
Se modifican los campos con el texto en blanco (“”)
Se modifica el estado de los campos, de “disable” a “enable”

Nombre: guardar_o_modificar_cliente()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información válida.
Post-condiciones: Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Actualiza los atributos del cliente con los datos validados.

Nombre: ingresar(clientes cliente)
Responsabilidades: Ingresa los datos del cliente a la base de datos.
Precondiciones: Existe la instancia de clientes.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement y lo asocia a cliente.
Se instancia pst con los datos de cliente.

- Modificar cliente

Nombre: mostrarListaClientes(String buscar)
Responsabilidades: Mostrar lista con los clientes registrados
Precondiciones:
Post-condiciones: se busca la lista de los objetos cliente

Nombre: modificarCliente(int fila)
Responsabilidades: Asigna a cada campo del formulario los datos de la fila (cliente) seleccionada en la tabla.
Precondiciones: Formulario debe restablecerse con valores “”.
Cliente debe estar registrado.
Post-condiciones: Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Se modifica el estado de los campos, de “disable” a “enable”.
Se instancia cliente con los valores recogidos del formulario.

Nombre: guardar_o_modificar_cliente()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información válida.
Post-condiciones: Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Actualiza los atributos del cliente con los datos validados.

Nombre: modificar(clientes cliente)
Responsabilidades: Actualiza los datos del cliente en la base de datos.
Precondiciones: Existe la instancia de clientes.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con los datos de cliente.
Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Se crea objeto modelo de tipo DefaultTableModel.

- Eliminar cliente

Nombre: mostrarListaClientes(String buscar)
Responsabilidades: Mostrar lista con los clientes registrados
Precondiciones:
Post-condiciones: Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Se crea objeto modelo de tipo DefaultTableModel.

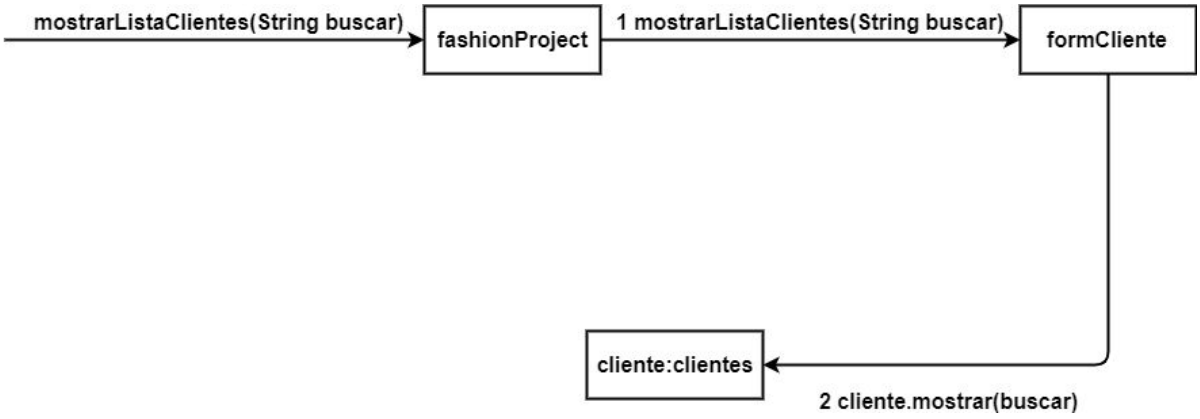
Nombre: eliminarCliente(int fila)
Responsabilidades: Detectar cliente a eliminar y mostrar mensaje de confirmación.
Precondiciones: Cliente debe estar registrado.
Post-condiciones: Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Se instancia cliente según los datos del id_cliente seleccionado.

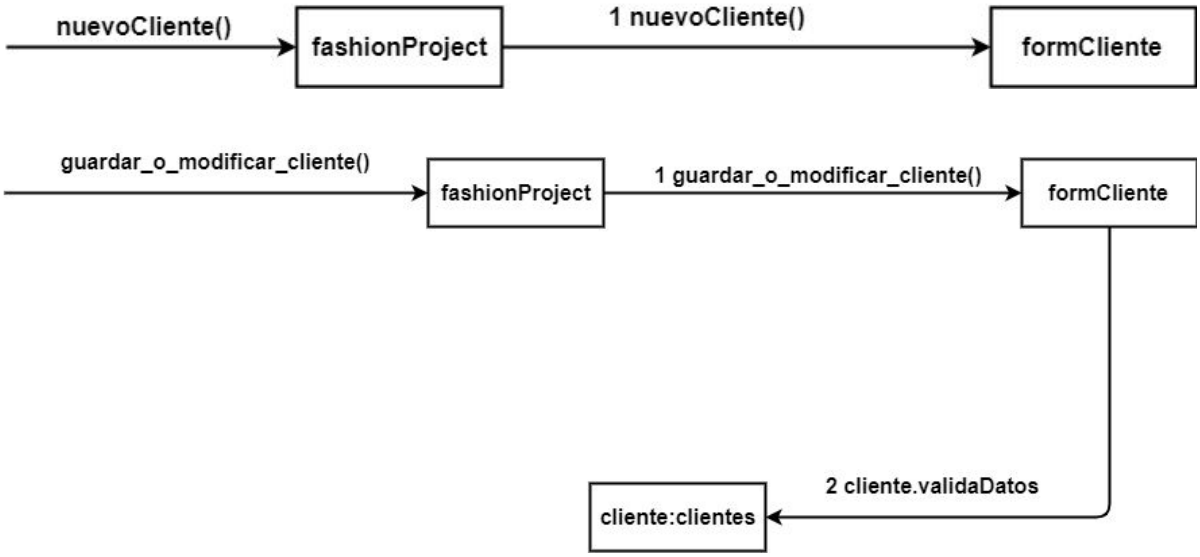
Nombre: eliminar(clientes cliente)
Responsabilidades: Cambiar de estado (actualizar) a cliente seleccionado (pasar de un estado eliminado a un estado “invisible”).
Precondiciones: Cliente debe estar registrado.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con el id_cliente del cliente a eliminar.

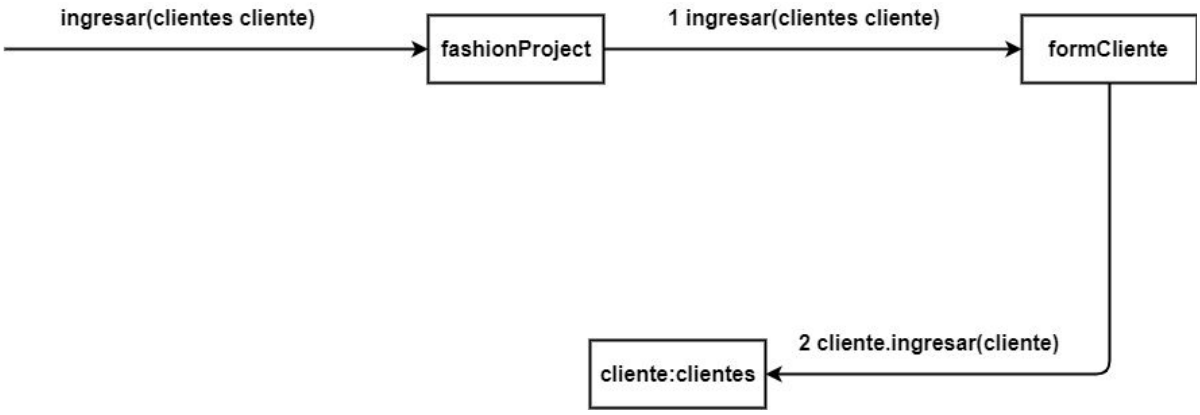
cliente.estado = 0 (estado de eliminado)
Clase formCliente() crea un objeto cliente y se asocia con formCliente().
Se crea objeto modelo de tipo DefaultTableModel.

Diagramas de Colaboración:

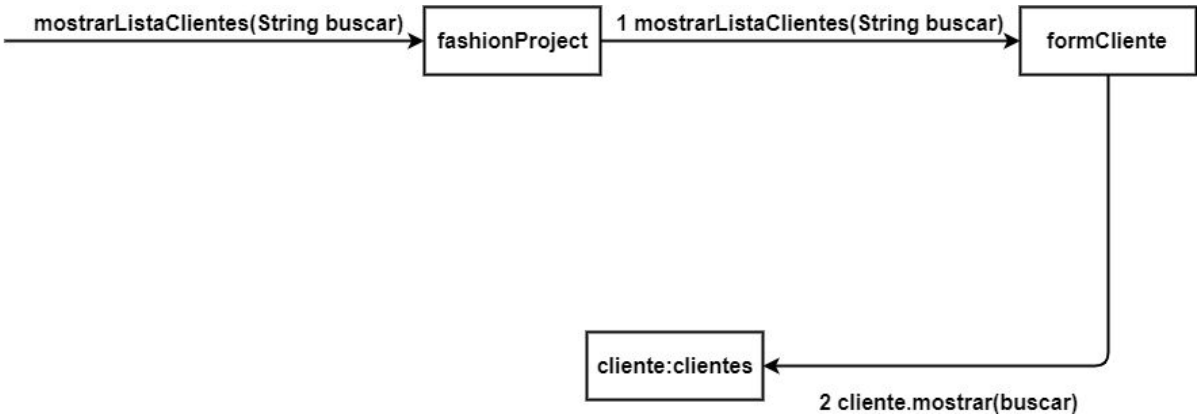
Ingresar cliente

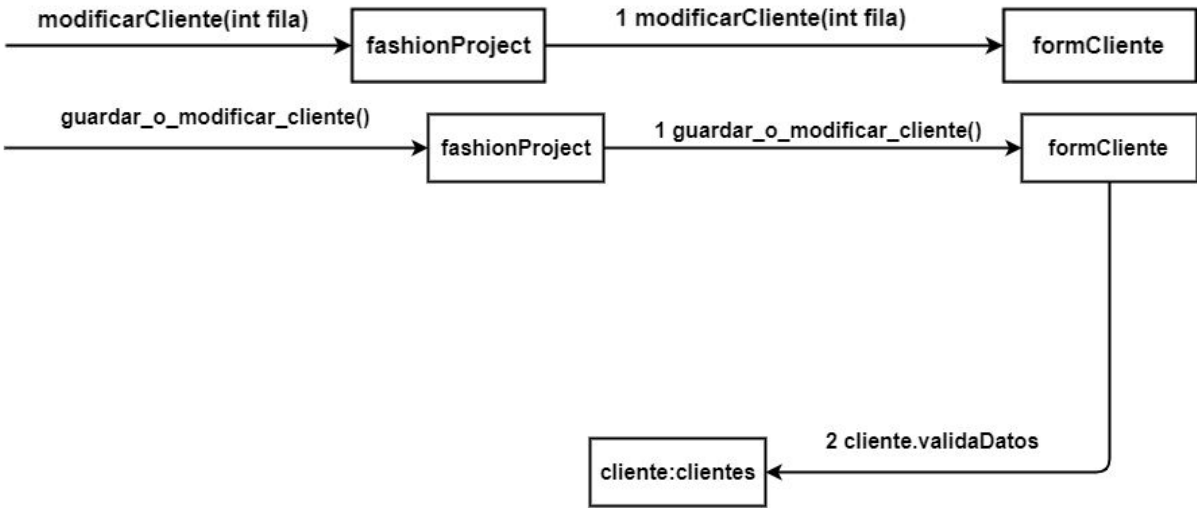


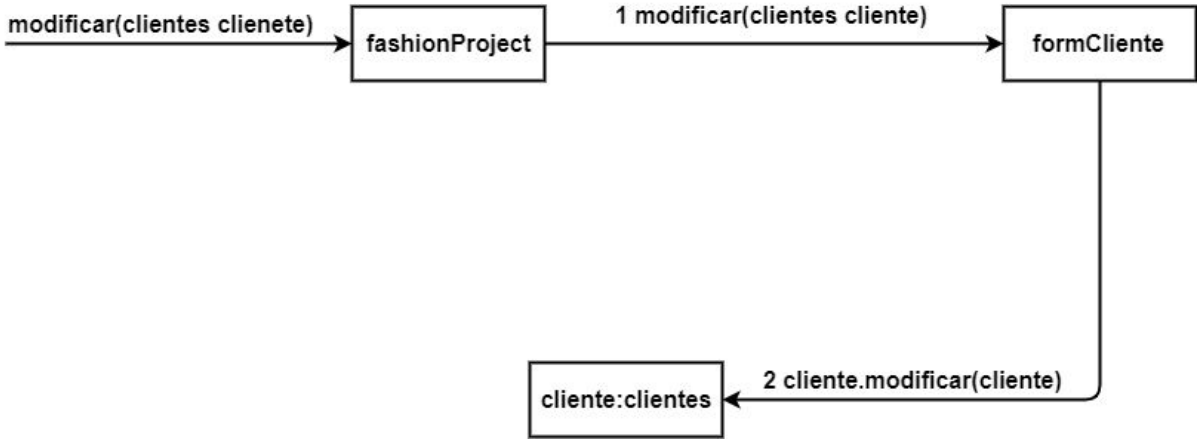




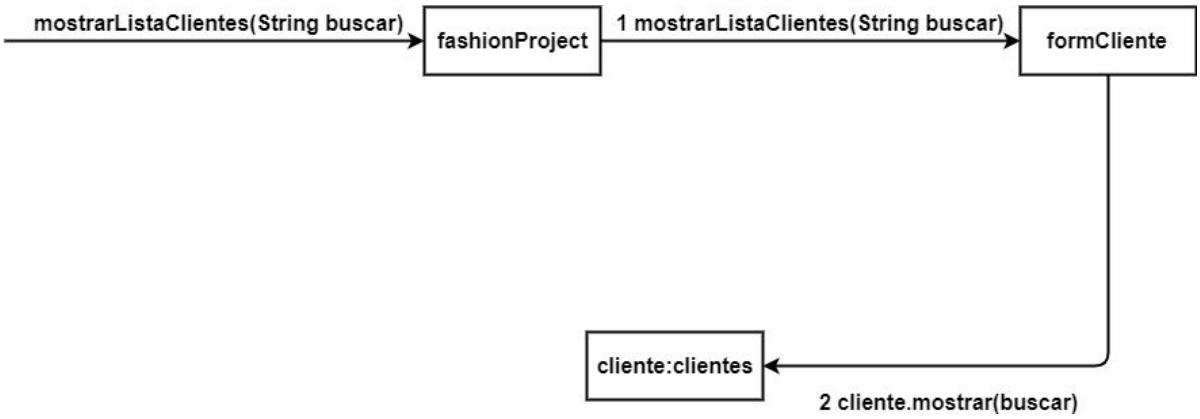
Modificar Cliente

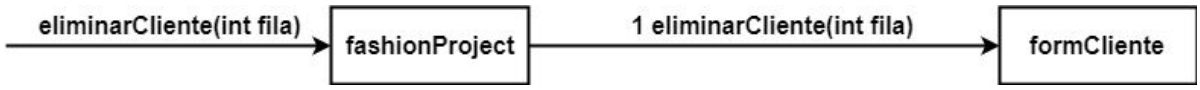






Eliminar Cliente





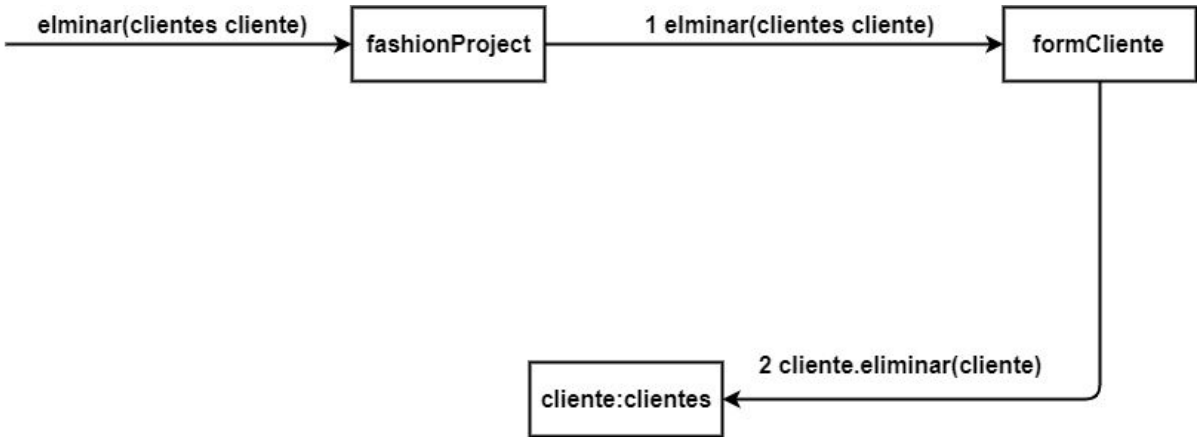


Diagrama de Estados:

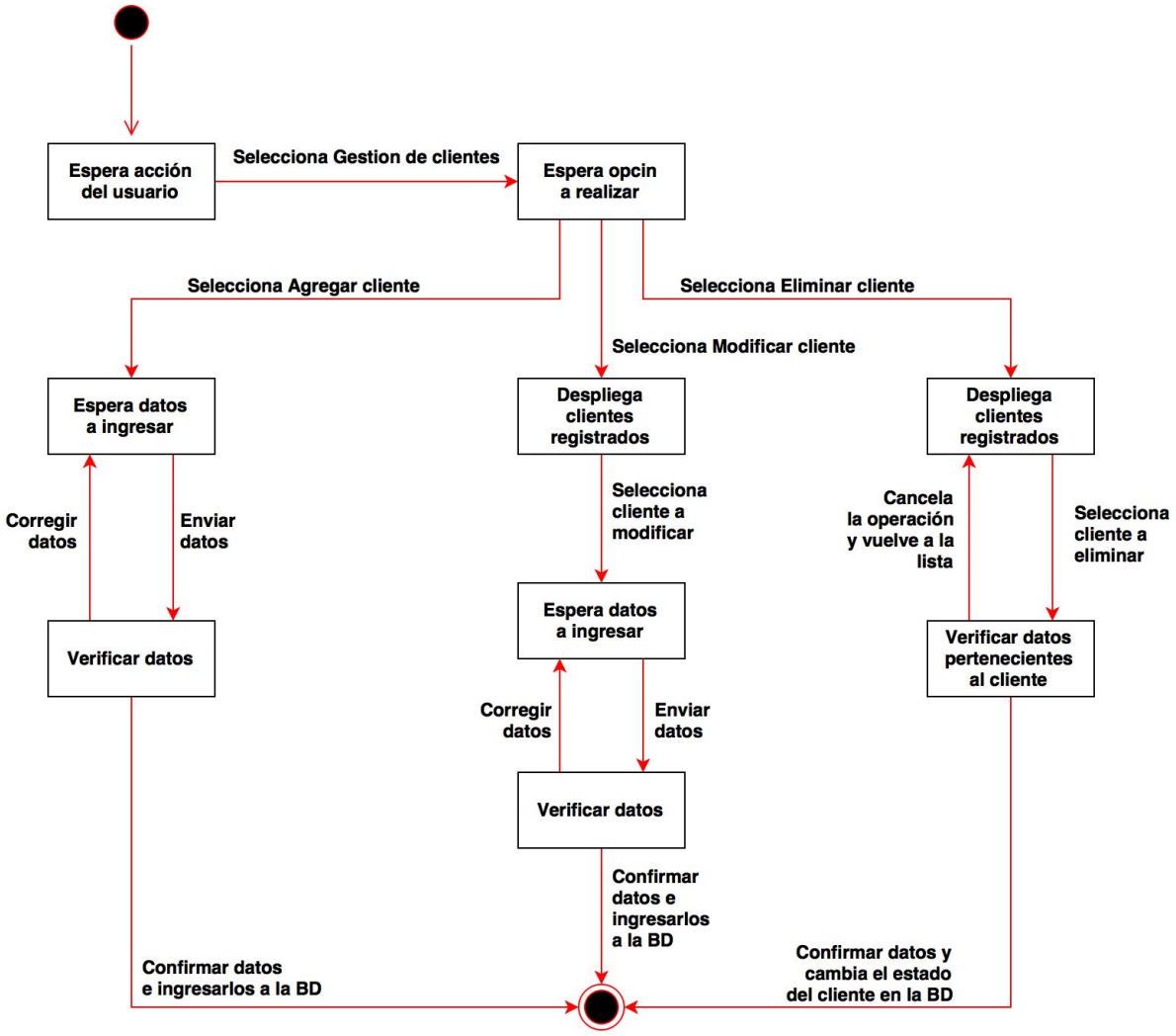
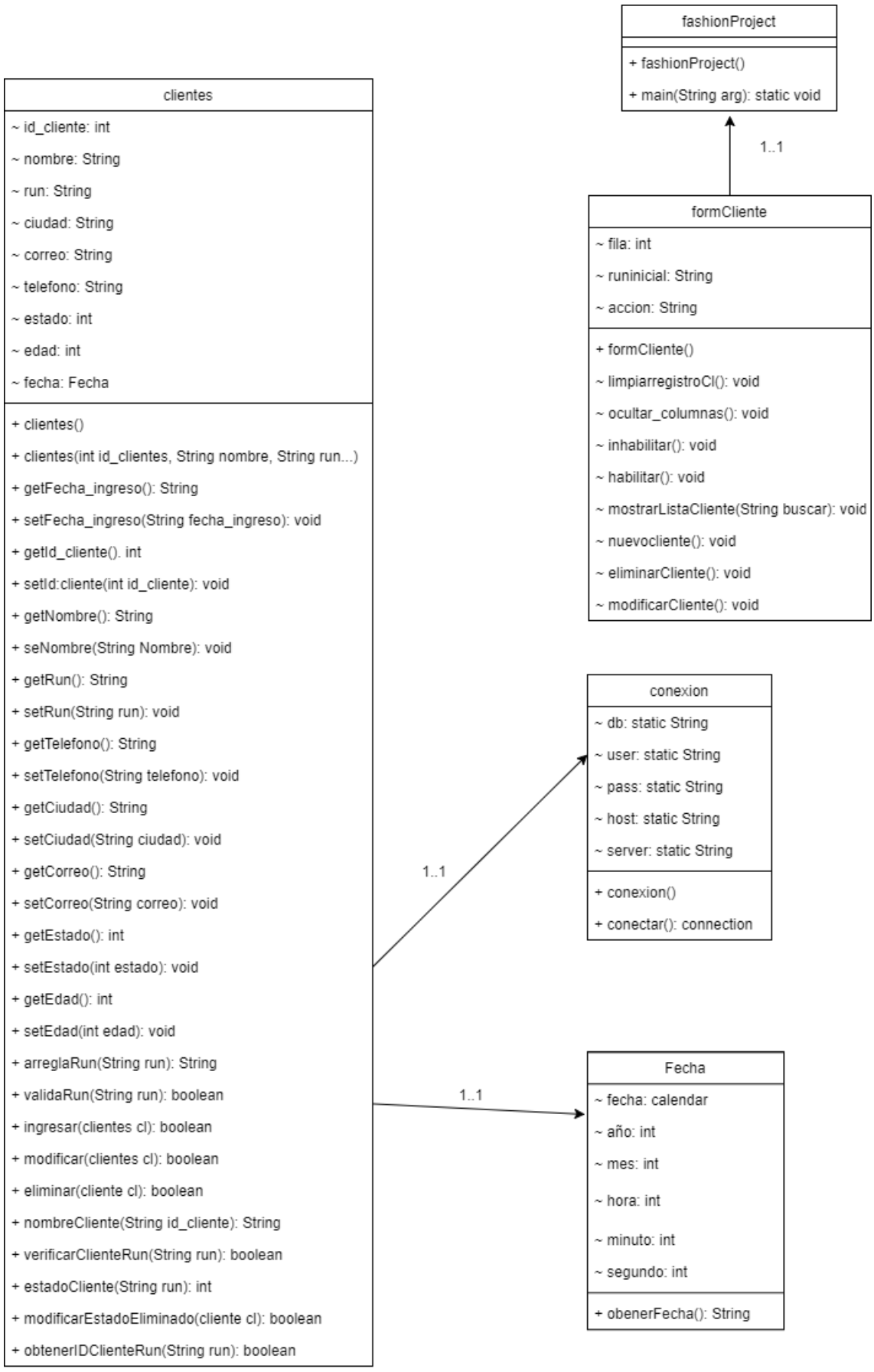


Diagrama de Clases:



Incremento 2:

Caso de uso: Gestión Trabajadores.

Actor: Usuario administrador.
Precondición: El usuario administrador debe estar registrado en el sistema.

OBS: para los cursos alternativos de modificar y "eliminar" trabajadores, estos deberán estar registrado en la base de datos.

Resumen: El usuario ingresa al sistema y gestiona a los trabajadores, es decir, puede ingresar un nuevo trabajador o también puede modificarlos y "eliminarlos".
Tipo: Primario - esencial

Curso normal de los eventos: Ingresar trabajador.

Actor	Sistema
1) El caso de uso comienza cuando el usuario administrador desea gestionar algún trabajador.	
2) El usuario administrador accede al sistema.	3) El sistema autoriza el ingreso del usuario.
	4) Sistema despliega el menú.
5) Usuario ingresa a la gestión de trabajadores.	6) Sistema despliega la ventana de gestión de trabajadores.
7) Usuario solicita los datos al trabajador y los ingresa.	8) Sistema verifica los datos ingresados.
	9) El trabajador es registrado, desplegando un mensaje.
	10) Muestra la lista actualizada de los trabajadores.
11) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: Trabajador ya existente en el sistema.

Actor	Sistema
	9*) El sistema despliega un mensaje diciendo que el trabajador ya se encuentra registrado.
10) Usuario corrige los datos o cancela la operación.	

Curso alternativo: Modificar trabajador.

Actor	Sistema
7*) Usuario administrador busca un trabajador al cual modificar sus datos.	8) Muestra al trabajador que coincide con la búsqueda.
9) Selecciona el trabajador.	10) Despliega los datos almacenados del trabajador.
11) Usuario administrador solicita datos al trabajador y los ingresa.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de los trabajadores.
14) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: Curso alternativo de “Modificar trabajador”.

Actor	Sistema
11**) Usuario solicita datos al trabajador y los ingresa de forma incompleta.	12) Despliega un mensaje de error.
13) Usuario completa todos los campos y los ingresa.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de los trabajadores.
14) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Eliminar trabajador.

Actor	Sistema
7*) Usuario busca un trabajador para eliminarlo.	8) Muestra al trabajador que coincide con la búsqueda.
9) Selecciona al trabajador.	10) Despliega los datos almacenados del trabajador.
11) Selecciona la opción de eliminar.	12) Despliega mensaje de confirmación.
13) Confirma su selección.	14) Cambia el estado del trabajador y no lo despliega en la lista de trabajadores.
	15) Se muestra la lista actualizada de los trabajadores.
16) El usuario realiza otra acción o sale del sistema.	

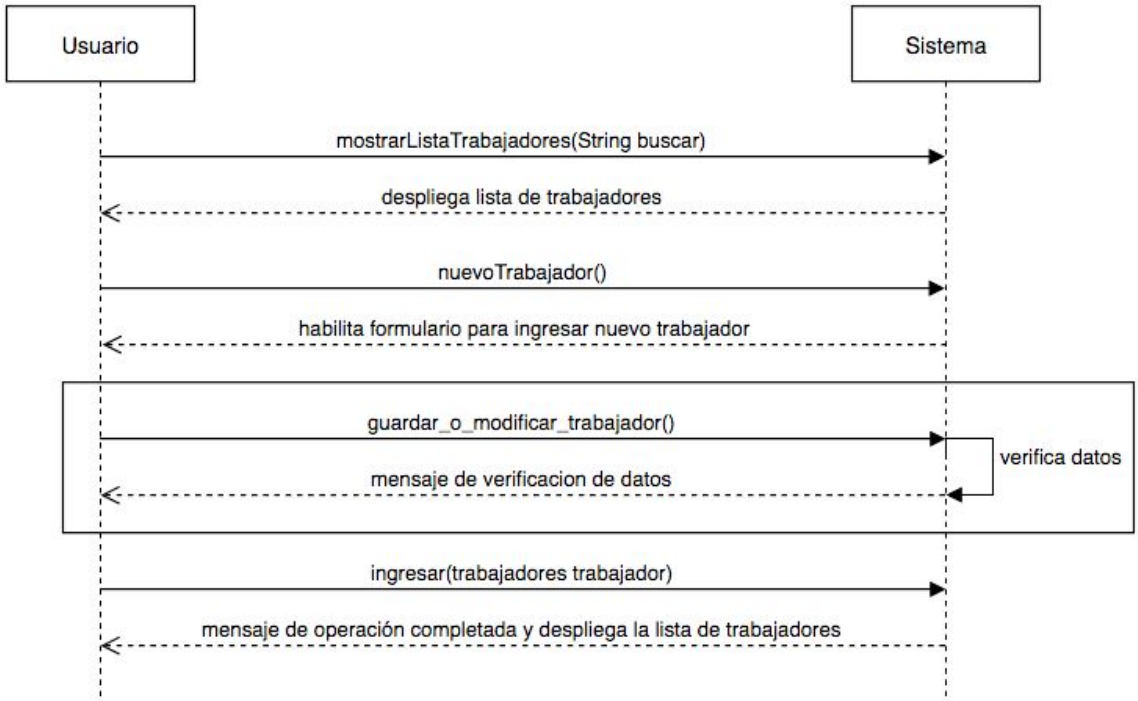
Curso alternativo: Curso alternativo de “Eliminar trabajador”.

Actor	Sistema

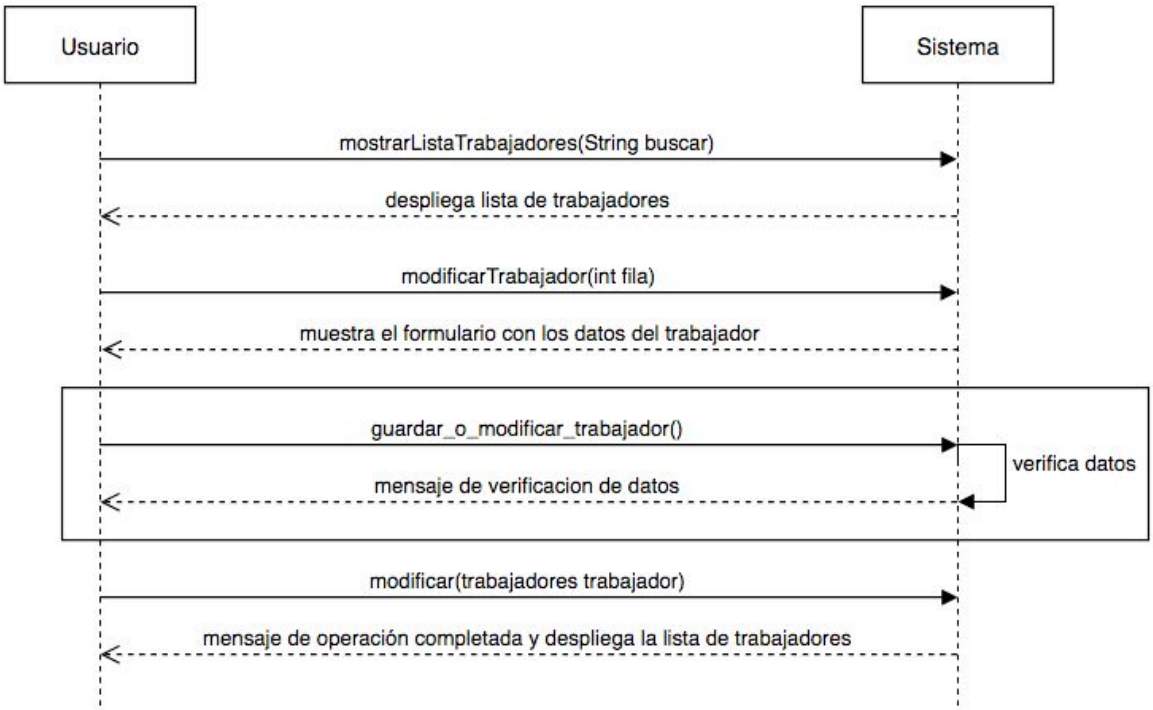
13**) Usuario se percata que seleccionó otro trabajador y cancela la operación.	

Diagramas de Secuencia:

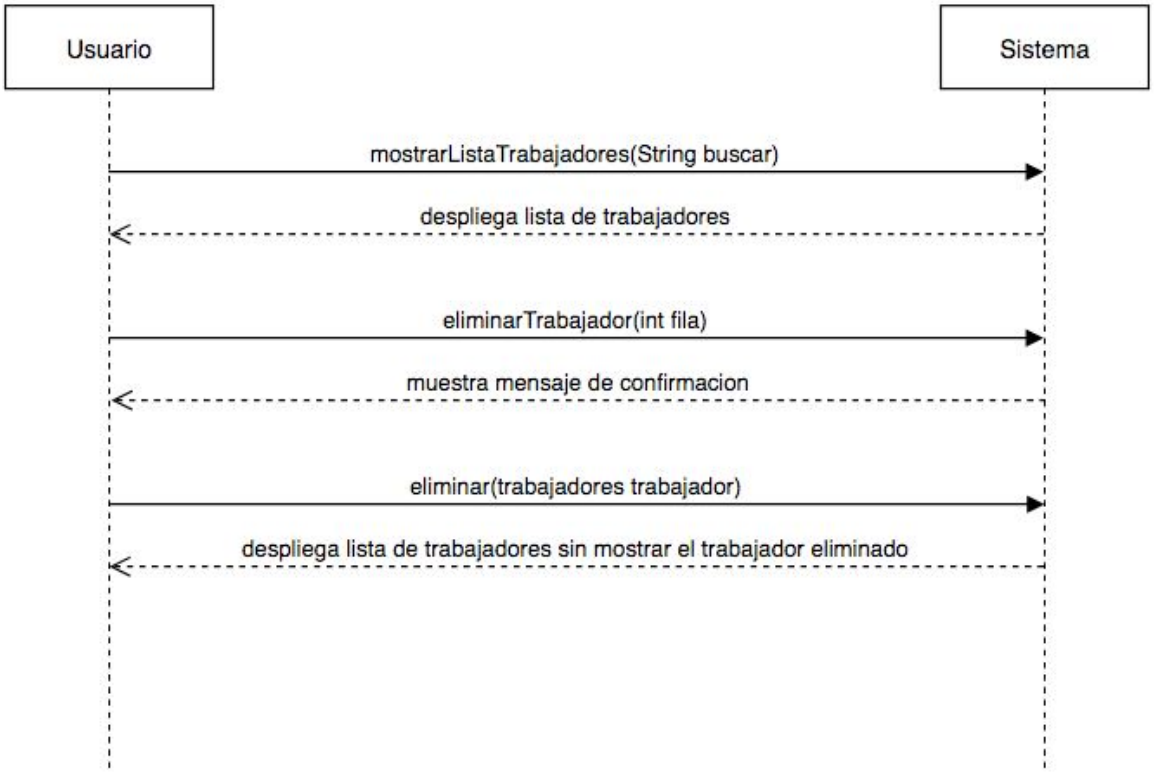
Ingresar Trabajador



Modificar Trabajador



Eliminar Trabajador



Contratos:

- Ingresar trabajador

Nombre: mostrarListaTrabajadores(String buscar)
Responsabilidades: Mostrar lista con los trabajadores registrados
Precondiciones:
Post-condiciones: Se crea objeto trabajador.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = trabajador.mostrar(buscar)
tabla.setModel(modelo)

Nombre: nuevoTrabajador()
Responsabilidades: Limpiar el formulario para luego habilitarlo para ingresar los datos.
Precondiciones:
Post-condiciones: Habilitar los campos para rellenar los datos del trabajador (setea los campos “” y los activa).

Nombre: guardar_o_modificar_trabajador()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información
Post-condiciones: Se crea objeto trabajadores.
Se instancia un trabajador de tipo trabajadores.

Nombre: ingresar(trabajador)
Responsabilidades: Ingresa los datos del trabajador a la base de datos.
Precondiciones: Existe la instancia de trabajadores.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con los datos de trabajador.

- Modificar trabajador

Nombre: mostrarListaTrabajadores(String buscar)
Responsabilidades: Mostrar lista con los trabajadores registrados
Precondiciones:
Post-condiciones: Se crea objeto trabajadores.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = trabajador.mostrar(buscar)
tabla.setModel(modelo)

Nombre: modificarTrabajador(int fila)
Responsabilidades: Asigna a cada campo del formulario los datos de la fila (trabajador) seleccionada en la tabla.
Precondiciones: Formulario debe restablecerse con valores “”.
Trabajador debe estar registrado.
Post-condiciones: Se crea objeto trabajador de tipo trabajadores.
Se instancia trabajador con los valores recogidos del formulario.
trabajador.datos.setText(tabla.getValueAt(fila, columnas tabla))

Nombre: guardar_o_modificar_trabajador()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información

Post-condiciones: Se crea objeto trabajadores.
Se instancia un trabajador de tipo trabajadores.

Nombre: modificar(trabajador)
Responsabilidades: Actualiza los datos del trabajador en la base de datos.
Precondiciones: Existe la instancia de trabajadores.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con los datos de trabajador.
Se crea objeto trabajadores.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = trabajador.mostrar(buscar)
tabla.setModel(modelo)

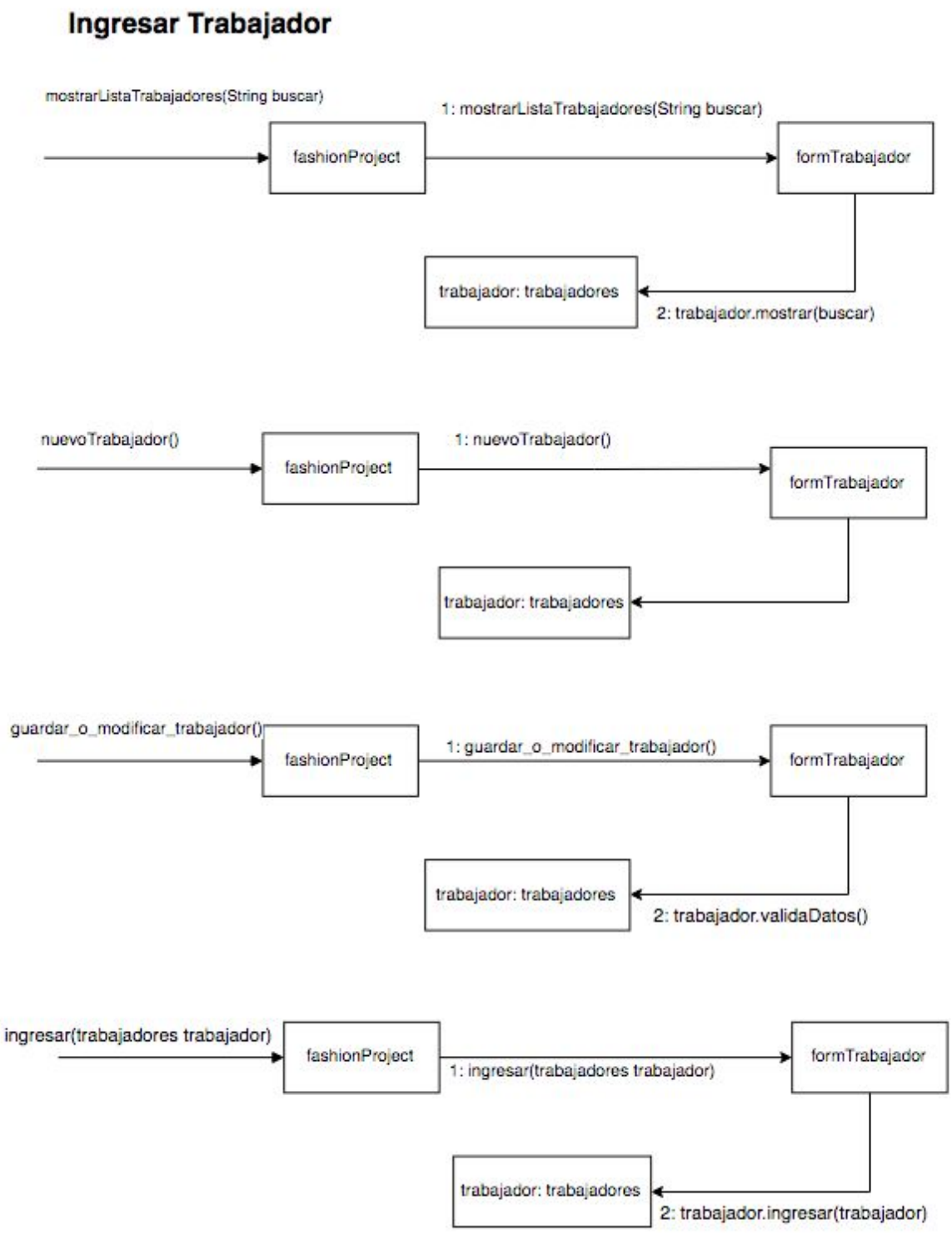
- Eliminar trabajador

Nombre: mostrarListaTrabajadores(String buscar)
Responsabilidades: Mostrar lista con los trabajadores registrados
Precondiciones:
Post-condiciones: Se crea objeto trabajadores.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = trabajador.mostrar(buscar)
tabla.setModel(modelo)

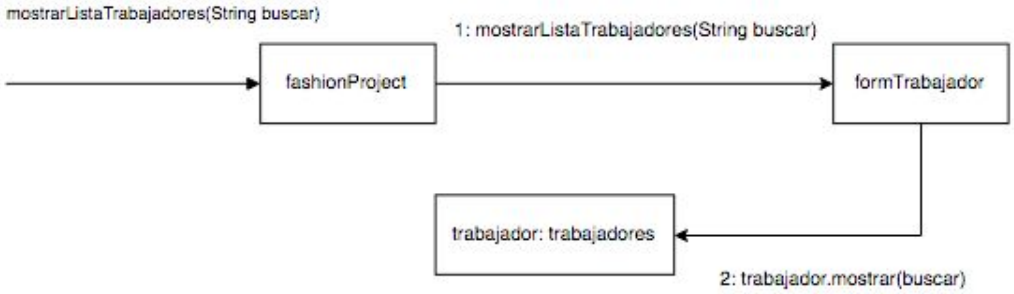
Nombre: eliminarTrabajador(int fila)
Responsabilidades: Detectar trabajador a eliminar y mostrar mensaje de confirmación.
Precondiciones: Trabajador debe estar registrado.
Post-condiciones: Se crea objeto trabajador de tipo trabajadores.
Se instancia trabajador según los datos del id_trabajador seleccionado.
trabajador.datos.setText(tabla.getValueAt(fila, columnas tabla))

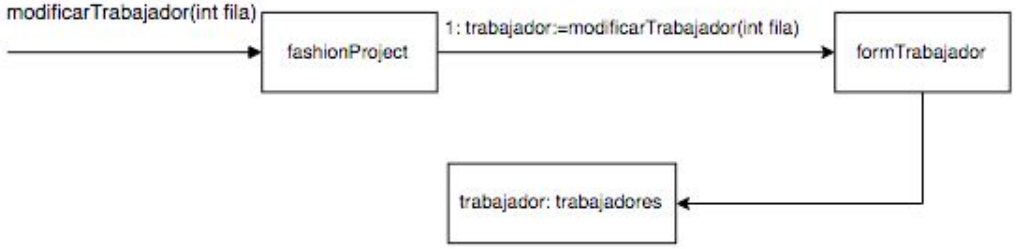
Nombre: eliminar(trabajadores trabajador)
Responsabilidades: Cambiar de estado (actualizar) a trabajador seleccionado (pasar de un estado eliminado a un estado “invisible”).
Precondiciones: Trabajador debe estar registrado.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con el id_trabajador del trabajador a eliminar.
trabajador.estado = 0 (estado de eliminado)
Se crea objeto trabajadores.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = trabajador.mostrar(buscar)
tabla.setModel(modelo)

Diagramas de Colaboración:



Modificar Trabajador





Eliminar Trabajador

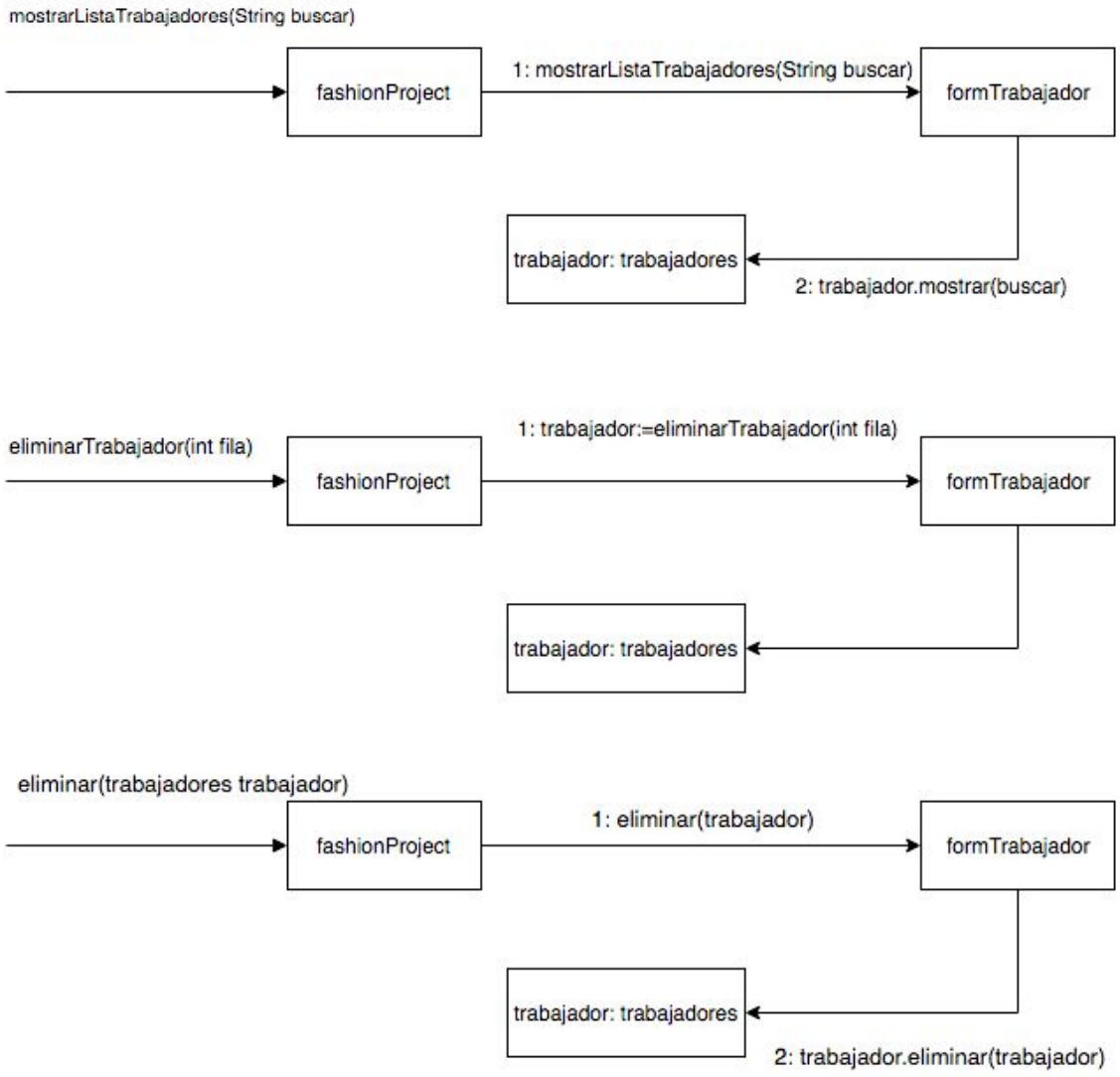
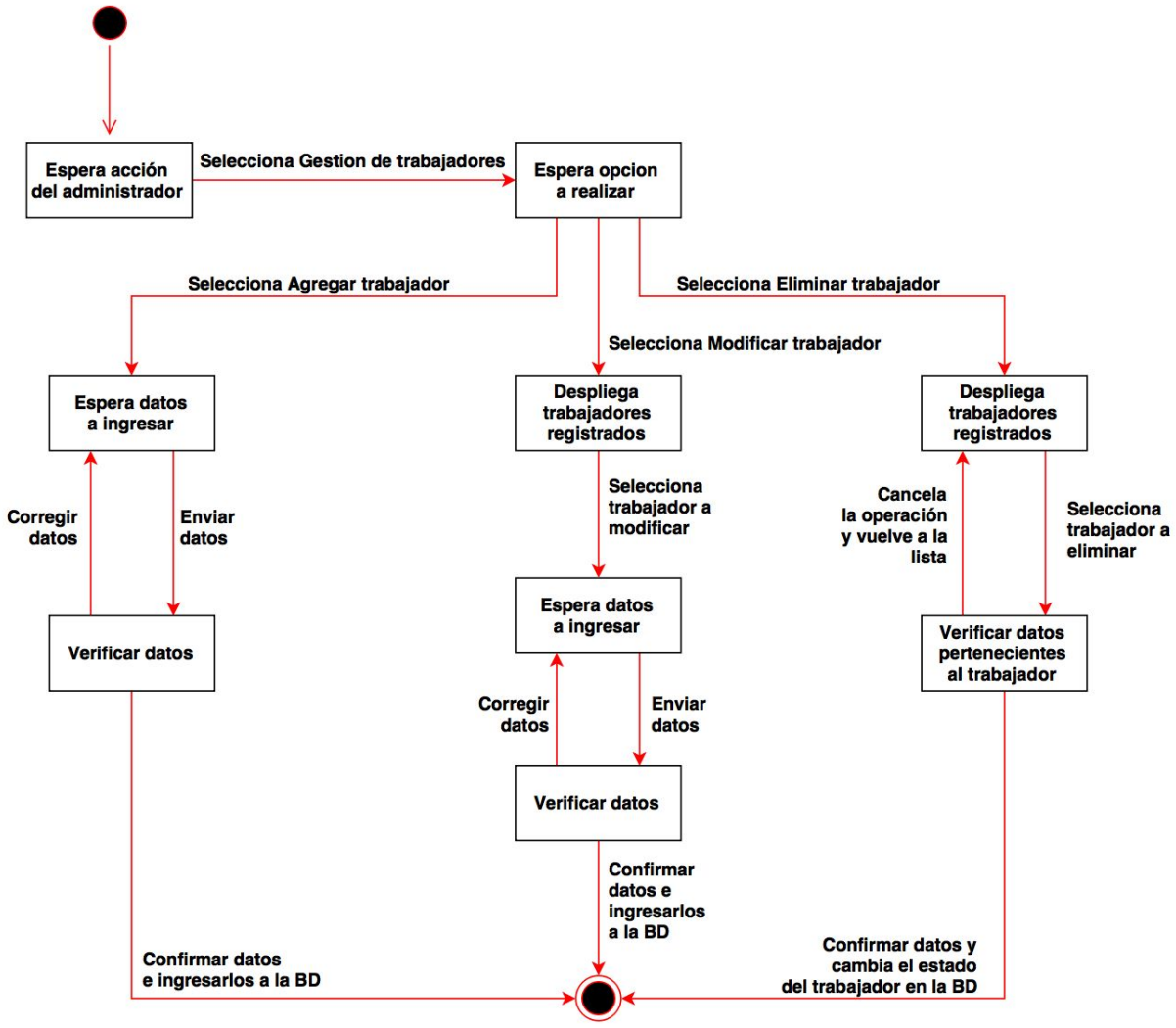
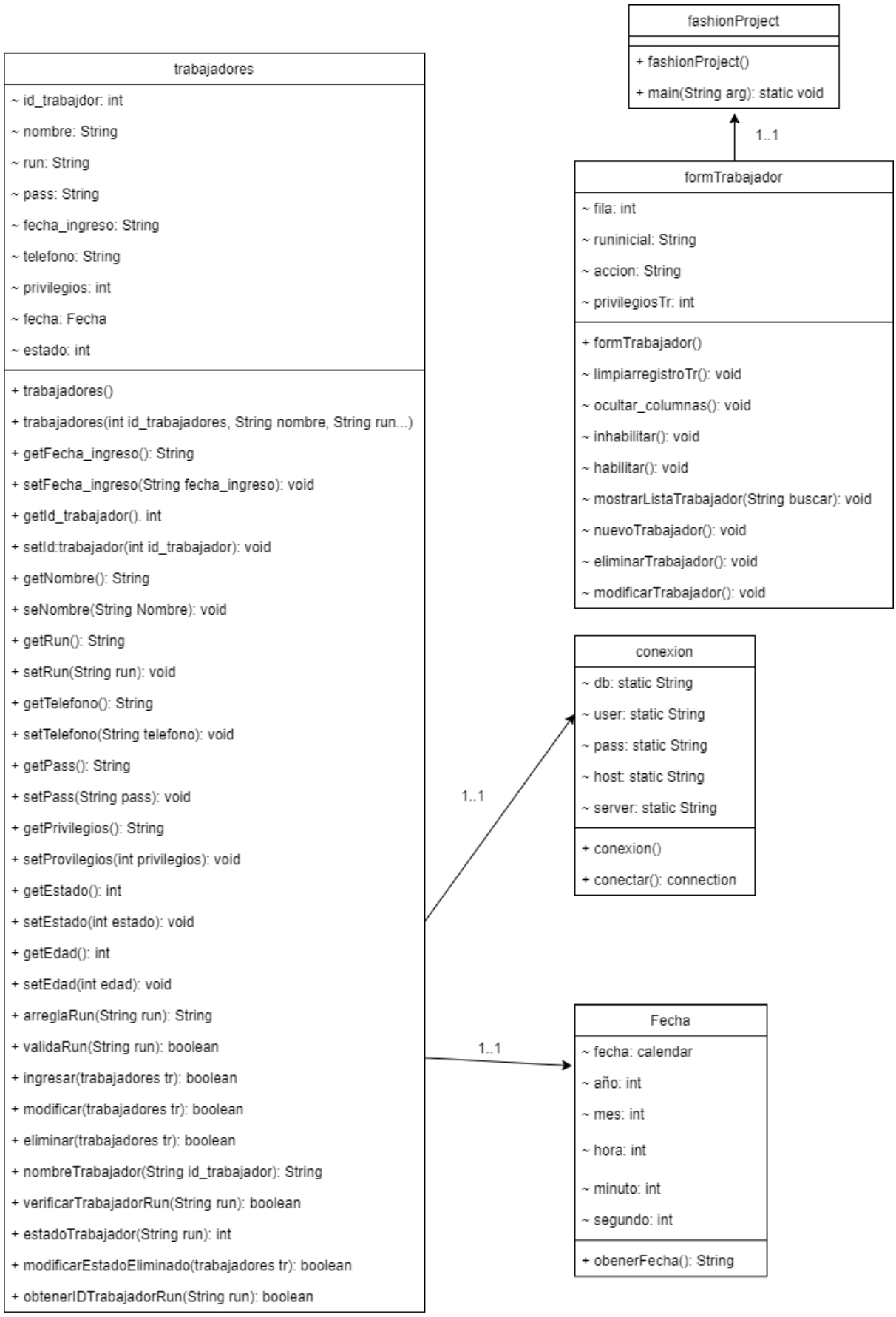


Diagrama de Estados:



* Los trabajadores son tratados con el nombre de "usuarios" dentro del sistema

Diagrama de clases:



Incremento 3:

Caso de uso: Gestión tratamientos.

Actor: Usuario administrador.
Precondición: El usuario administrador debe estar registrado en el sistema.

OBS: para los cursos alternativos de modificar y "eliminar" tratamientos, estos deberán estar registrados en la base de datos.

Resumen: El usuario ingresa al sistema y gestiona los tratamientos, es decir, puede ingresar un nuevo tratamiento o también puede modificarlos y "eliminarlos".
Tipo: Primario - esencial

Curso normal de los eventos: Ingresar tratamiento.

Actor	Sistema
1) El caso de uso comienza cuando el usuario administrador desea gestionar algún tratamiento.	
2) El usuario administrador accede al sistema.	3) El sistema autoriza el ingreso del usuario.
	4) Sistema despliega el menú.
5) Usuario ingresa a la gestión de tratamientos.	6) Sistema despliega la ventana de gestión de tratamientos.
7) Usuario ingresa los datos del tratamiento.	8) Sistema verifica los datos ingresados.
	9) El tratamiento es registrado, desplegando un mensaje.
	10) Muestra la lista actualizada de los tratamientos.
11) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: tratamiento ya existente en el sistema.

Actor	Sistema
	9*) El sistema despliega un mensaje diciendo que el tratamiento ya se encuentra registrado.
10) Usuario corrige los datos o cancela la operación.	

Curso alternativo: Modificar tratamiento.

Actor	Sistema
7*) Usuario administrador busca un tratamiento al cual modificar sus datos.	8) Muestra al tratamiento que coincide con la búsqueda.
9) Selecciona el tratamiento.	10) Despliega los datos almacenados del tratamiento.
11) Usuario administrador ingresa los datos del tratamiento.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de los tratamientos.
14) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: Curso alternativo de “Modificar tratamiento”.

Actor	Sistema
11**) Usuario ingresa datos del tratamiento los ingresa de forma incompleta.	12) Despliega un mensaje de error.
13) Usuario completa todos los campos y los ingresa.	12) Se modifican los datos, desplegando un mensaje.
	13) Muestra la lista actualizada de los tratamientos.
14) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Eliminar tratamiento.

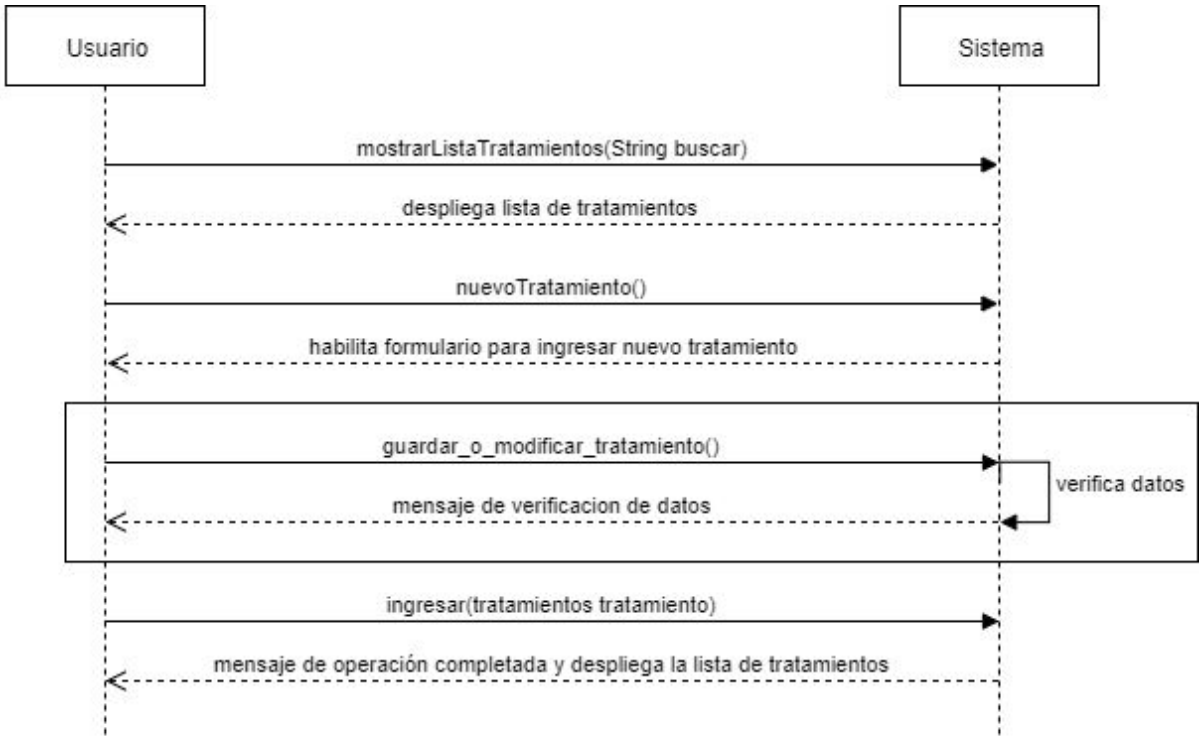
Actor	Sistema
7*) Usuario busca un tratamiento para eliminarlo.	8) Muestra el tratamiento que coincide con la búsqueda.
9) Selecciona un tratamiento.	10) Despliega los datos almacenados del tratamiento.
11) Selecciona la opción de eliminar.	12) Despliega mensaje de confirmación.
13) Confirma su selección.	14) Cambia el estado del tratamiento y no lo despliega en la lista de tratamientos.
	15) Se muestra la lista actualizada de los tratamientos.
16) El usuario realiza otra acción o sale del sistema.	

Curso alternativo: Curso alternativo de “Eliminar tratamiento”.

Actor	Sistema
13**) Usuario se percata que seleccionó otro tratamiento y cancela la operación.	

Diagramas de Secuencia:

Ingresar tratamiento



Modificar tratamiento

Eliminar tratamiento

Contratos:

- Ingresar tratamiento

Nombre: mostrarListaTratamientos(String buscar)
Responsabilidades: Mostrar lista con los tratamientos registrados
Precondiciones:
Post-condiciones: se busca la lista de los objetos tratamiento

Nombre: nuevoTratamiento()
Responsabilidades: Limpiar el formulario para luego habilitarlo para ingresar los datos.
Precondiciones:
Post-condiciones: Habilitar los campos para rellenar los datos del tratamiento(setea los campos “” y los activa).

Nombre: guardar_o_modificar_tratamiento()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información
Post-condiciones: Se crea objeto tratamientos.
Se instancia un tratamiento de tipo tratamientos.

Nombre: ingresar(tratamientos tratamiento)
Responsabilidades: Ingresa los datos del tratamiento a la base de datos.
Precondiciones: Existe la instancia de tratamientos.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con los datos de tratamiento.

- Modificar tratamiento

Nombre: mostrarListaTratamientos(String buscar)
Responsabilidades: Mostrar lista con los tratamientos registrados
Precondiciones:
Post-condiciones: Se crea objeto tratamientos.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = tratamiento.mostrar(buscar)
tabla.setModel(modelo)

Nombre: modificarTratamiento(int fila)
Responsabilidades: Asigna a cada campo del formulario los datos de la fila (tratamiento) seleccionada en la tabla.
Precondiciones: Formulario debe restablecerse con valores “”.
Tratamiento debe estar registrado.
Post-condiciones: Se crea objeto tratamiento de tipo tratamientos.
Se instancia tratamiento con los valores recogidos del formulario.
tratamiento.datos.setText(tabla.getValueAt(fila, columnas tabla))

Nombre: guardar_o_modificar_tratamiento()
Responsabilidades: Verifica y valida los datos ingresados por el usuario.
Precondiciones: Los campos obligatorios están con información
Post-condiciones: Se crea objeto tratamientos.
Se instancia un tratamiento de tipo tratamientos.

Nombre: modificar(tratamiento)
Responsabilidades: Actualiza los datos del tratamiento en la base de datos.
Precondiciones: Existe la instancia de tratamiento.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con los datos del tratamiento.
Se crea objeto tratamiento.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = tratamiento.mostrar(buscar)
tabla.setModel(modelo)

- Eliminar tratamiento

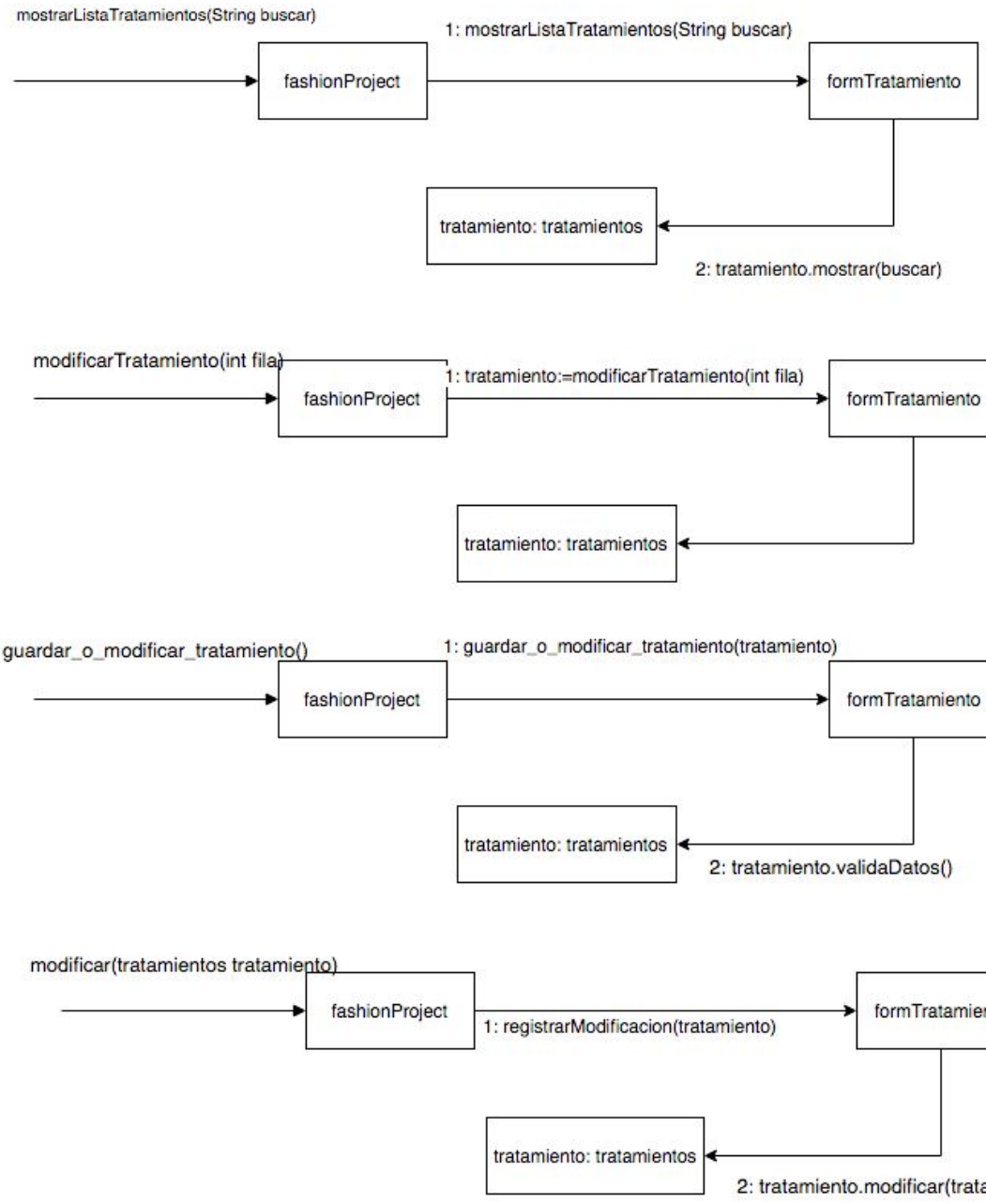
Nombre: mostrarListaTratamientos(String buscar)
Responsabilidades: Mostrar lista con los tratamiento registrados
Precondiciones:
Post-condiciones: Se crea objeto tratamientos.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = tratamiento.mostrar(buscar)
tabla.setModel(modelo)

Nombre: eliminarTratamiento(int fila)
Responsabilidades: Detectar tratamiento a eliminar y mostrar mensaje de confirmación.
Precondiciones: Tratamiento debe estar registrado.
Post-condiciones: Se crea objeto tratamiento de tipo tratamientos.
Se instancia tratamiento según los datos del id_tratamiento seleccionado.
tratamiento.datos.setText(tabla.getValueAt(fila, columnas tabla))

Nombre: eliminar(tratamientos tratamiento)
Responsabilidades: Cambiar de estado (actualizar) a tratamiento seleccionado (pasar de un estado eliminado a un estado “invisible”).
Precondiciones: Tratamiento debe estar registrado.
Post-condiciones: Se crea objeto pst de tipo PreparedStatement.
Se instancia pst con el id_tratamiento del tratamiento a eliminar.
tratamiento.estado = 0 (estado de eliminado)
Se crea objeto tratamiento.
Se crea objeto modelo de tipo DefaultTableModel.
modelo = tratamiento.mostrar(buscar)
tabla.setModel(modelo)

Diagrama de Colaboración:

Modificar Tratamiento



Eliminar Tratamiento

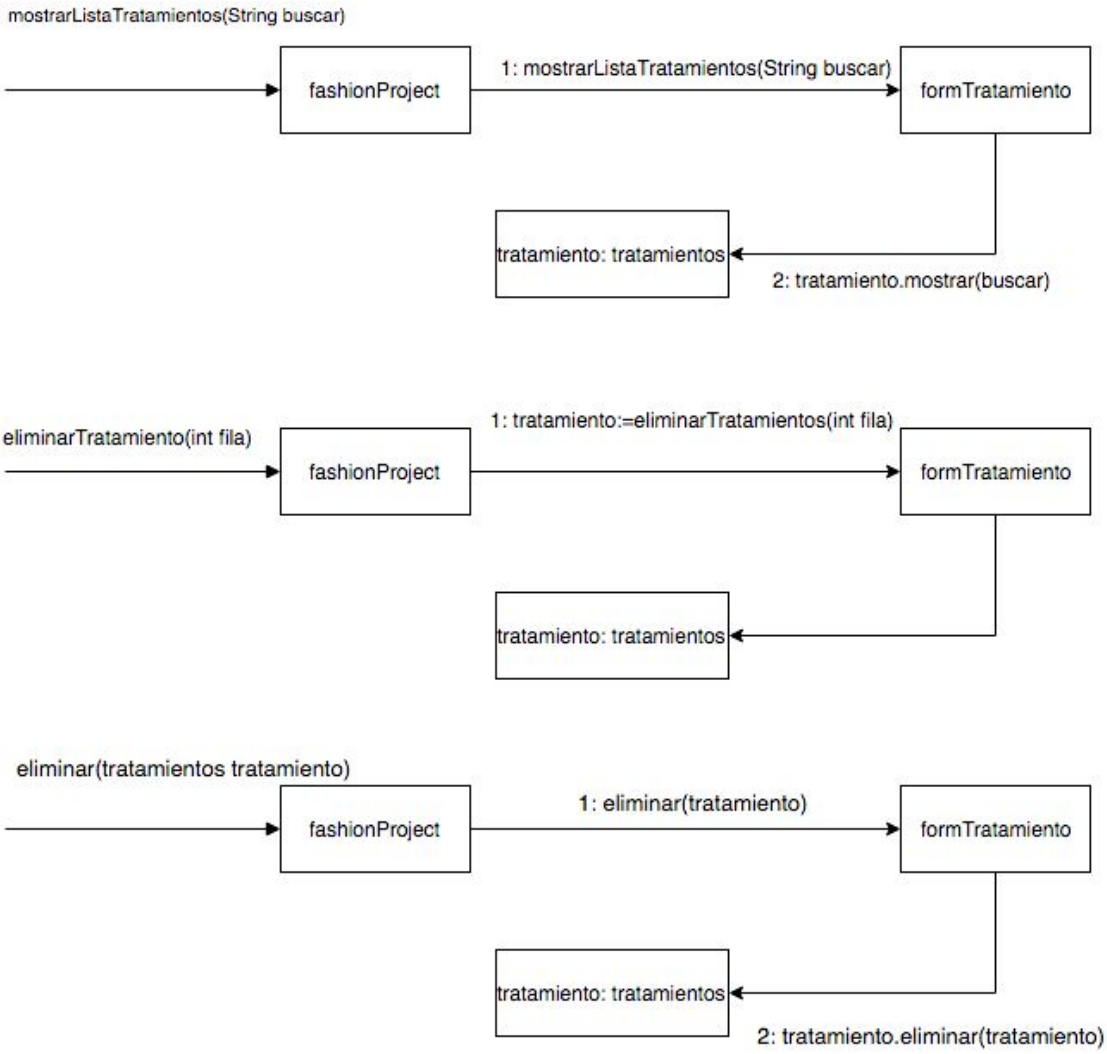


Diagrama de Estados:

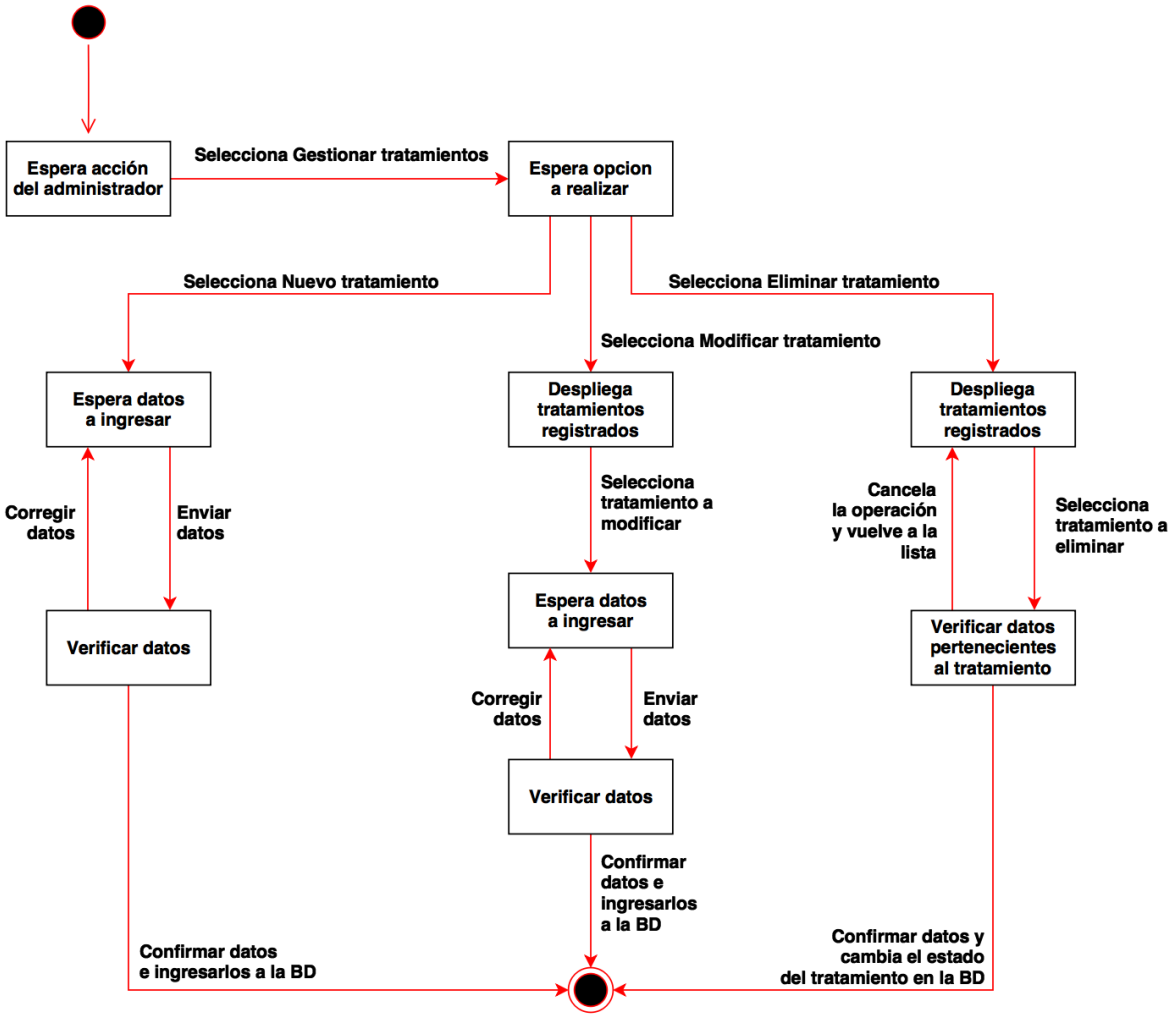
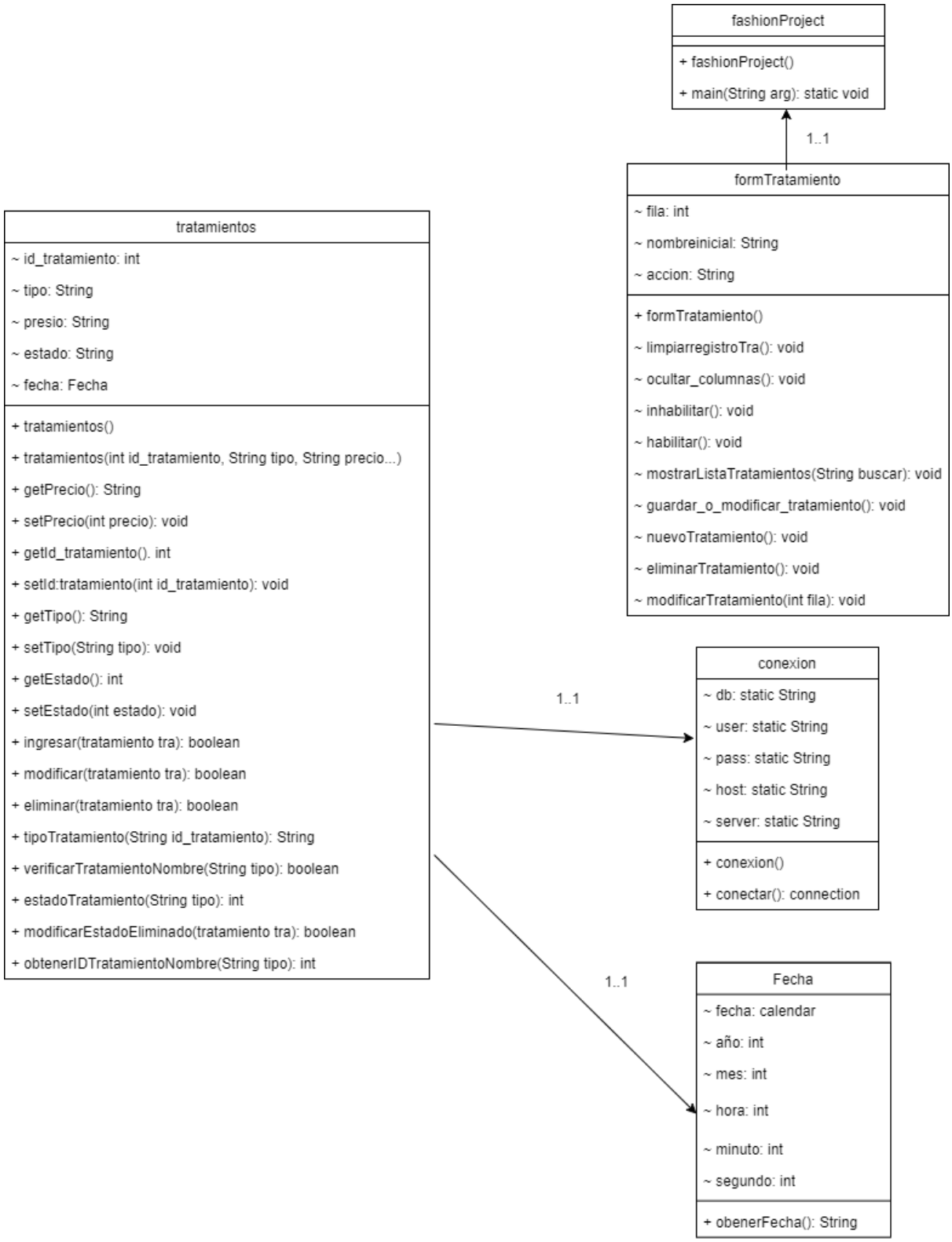


Diagrama de clases:



Incremento 4:

Caso de uso: Registrar Venta.

Actor: Usuario administrador, usuario normal.

Precondición: Los usuarios deben estar registrado en el sistema.
El/los tratamiento deben estar registrados y disponibles.

Resumen: Luego de la atención al cliente por parte del usuario, se gestiona y registra el tratamiento al cliente, es decir, se lleva a cabo el registro de la venta (servicio realizado).

Tipo: Primario - esencial

Curso normal de los eventos: Registrar Venta - Cliente registrado.

Actor	Sistema
1) El caso de uso comienza cuando el usuario desea registrar una venta.	
2) El usuario accede al sistema.	3) El sistema autoriza el ingreso del usuario.
	4) Sistema despliega el menú.
5) Usuario ingresa a Registrar Venta.	6) Sistema despliega la ventana para registrar una venta.
7) Usuario selecciona los datos del cliente y del tratamiento realizado.	8) Sistema verifica y obtiene datos respectivos de los tratamientos y cliente y lo despliega.
	9) Sistema indica información resumen de la venta, junto con el monto a pagar.
10) Usuario acepta el pago y finaliza la operación y registro de venta.	11) Sistema muestra un mensaje de que la operación se ha realizado con éxito.
12) Usuario decide si cerrar sesion o si desea registrar otra venta.	

Curso alternativo de los eventos: Registrar Venta - Cliente no registrado y agrega uno nuevo.

Actor	Sistema
* 7) Usuario se percata de que el cliente no está registrado.	
8) Usuario selecciona la opción agregar	9) Sistema despliega una ventana de registro de cliente desde la misma ventana de registro de venta.

10) Usuario solicita los datos y realiza el registro del nuevo cliente.	11) Sistema verifica que los datos y despliega un mensaje.
12) Usuario selecciona los datos del cliente y del tratamiento realizado.	13) Sistema verifica y obtiene datos respectivos de los tratamientos y clientes y lo despliega.
	14) Sistema indica información resumen de la venta, junto con el monto a pagar.
15) Usuario acepta el pago y finaliza la operación y registro de venta.	16) Sistema muestra un mensaje de que la operación se ha realizado con éxito.
17) Usuario decide si cerrar sesion o si desea registrar otra venta.	

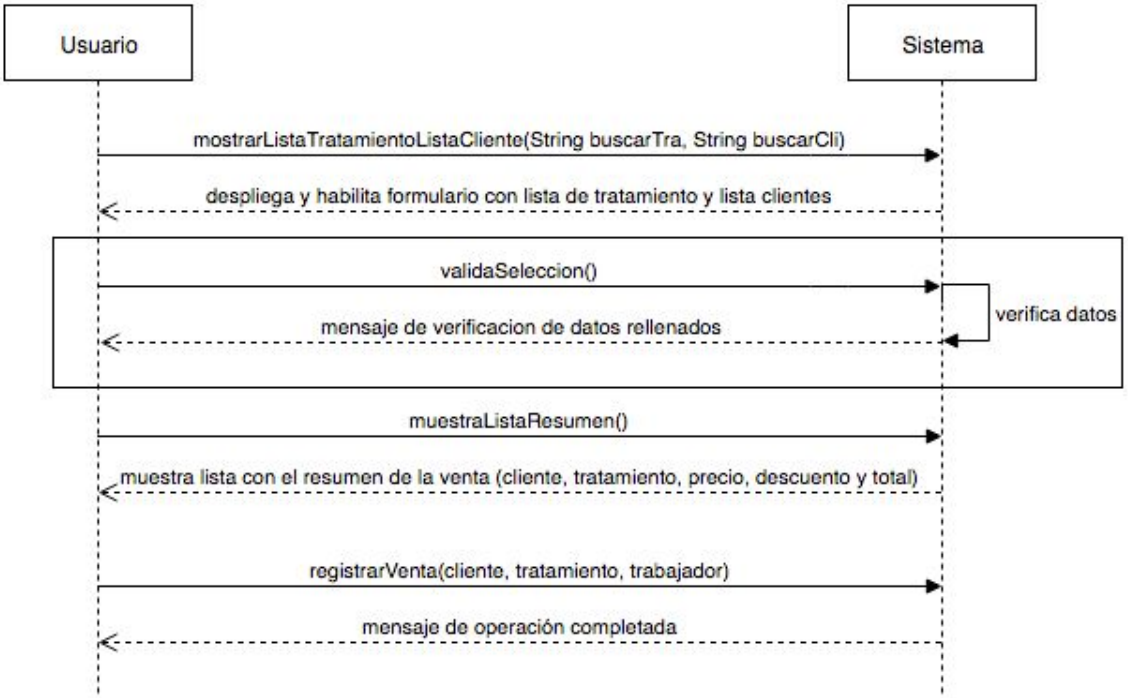
Curso alternativo de los eventos: Registrar Venta - Cliente no registrado y lo deja como anónimo.

Actor	Sistema
* 7) Usuario se percata de que el cliente no está registrado.	
8) Usuario selecciona cliente anónimo de la lista clientes	9) Sistema indica información resumen de la venta, junto con el monto a pagar.
10) Usuario acepta el pago y finaliza la operación y registro de venta.	11) Sistema muestra un mensaje de que la operación se ha realizado con éxito.
12) Usuario decide si cerrar sesion o si desea registrar otra venta.	

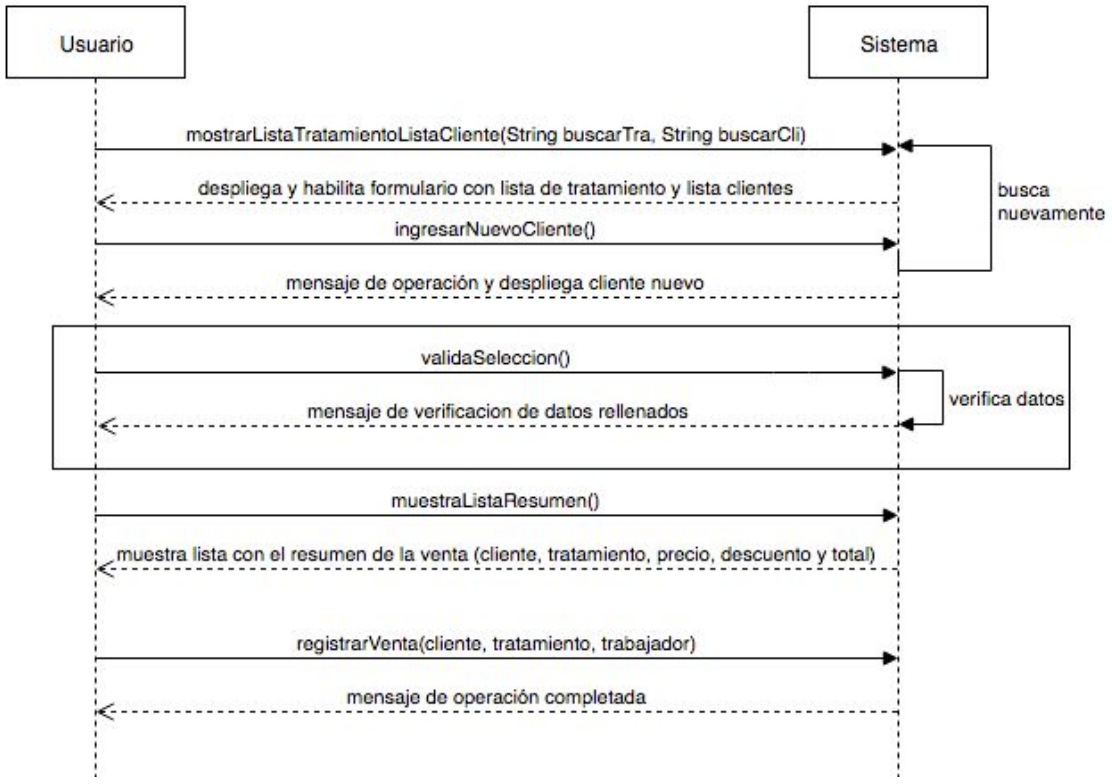
Diagramas de Secuencia:

https://drive.google.com/file/d/0B0Ns_S1eFY6sYXFKZktiRXc0OXM/view?usp=sharing

Registrar Venta (Cliente ya registrado en el sistema)



Registrar Venta (Cliente sin registrar en el sistema y agrega uno nuevo)



Contratos:

- Registrar venta (cliente ya registrado en el sistema)

Nombre: mostrarListaTratamientoListaCliente(String buscarTra, String buscarCli)
Responsabilidades: muestra la lista de los clientes y la de los tratamientos
Precondiciones: el cliente y el tratamiento debe estar registrado en el sistema
usuario debe estar registrado y autenticado en el sistema
Post-condiciones: se busca la lista de los objetos cliente
se busca la lista de los objetos tratamiento

Nombre: validaSeleccion()
Responsabilidades: verifica que se seleccione al menos un tratamiento y un cliente.
Precondiciones:
Post-condiciones: al objeto cliente se le asocia uno o más objetos tratamiento

Nombre: muestraListaResumen()
Responsabilidades: despliega una lista con los datos nombre del cliente, tipo de tratamiento, precio del tratamiento y si tiene algún tipo de descuento, además del total a pagar.
Precondiciones: en la primera iteración los atributos deben estar formateados
Post-condiciones:

Nombre: registrarVenta(cliente, tratamiento, trabajador)
Responsabilidades: crear un objeto venta y lo asigna al historial de ventas
Precondiciones: existe relación cliente-tratamiento
Post-condiciones: Se crea objeto venta_total de tipo venta.
Se instancia venta_total con los datos de cliente, tratamiento y trabajador en el historial de ventas
El atributo total de la clase venta se modifica con respecto a los atributos “precio” de la clase tratamiento y “descuento” de la clase cliente.

- Registrar venta (cliente sin registrar y agregar nuevo)

Nombre: mostrarListaTratamientoListaCliente(String buscarTra, String buscarCli)
Responsabilidades: muestra la lista de los clientes y la de los tratamientos
Precondiciones: el cliente y el tratamiento debe estar registrado en el sistema
usuario debe estar registrado y autenticado en el sistema
Post-condiciones: se busca la lista de los objetos cliente
se busca la lista de los objetos tratamiento

Nombre: agregarCliente()
Responsabilidades: dar la opción de agregar un cliente nuevo
Precondiciones: no se encontró instancia del objeto cliente con los datos buscados
Post-condiciones: se invoca la clase gestión clientes
se crea objeto cliente
se asignan los datos a los atributos del objeto cliente

Nombre: validaSeleccion()
Responsabilidades: verifica que se seleccione al menos un tratamiento y un cliente.
Precondiciones:
Post-condiciones: al objeto cliente se le asocia uno o más objetos tratamiento

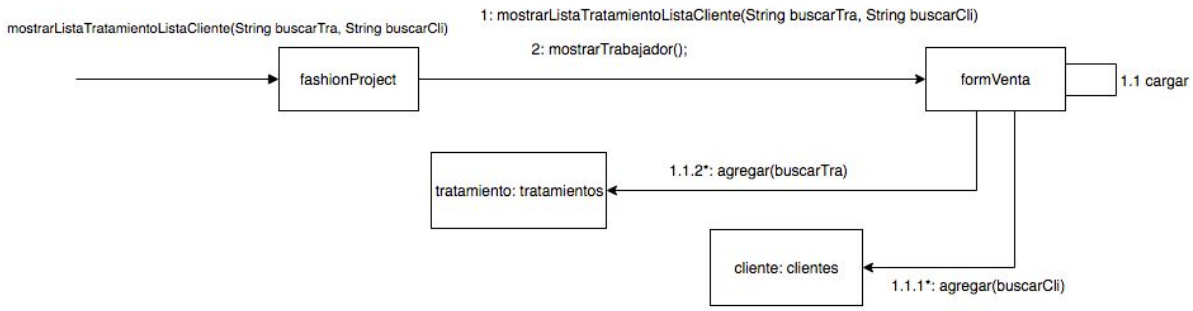
Nombre: muestraListaResumen()
Responsabilidades: despliega una lista con los datos nombre del cliente, tipo de tratamiento, precio del tratamiento y si tiene algún tipo de descuento, además del total a pagar.
Precondiciones: en la primera iteración los atributos deben estar formateados
Post-condiciones:

Nombre: registrarVenta(cliente, tratamiento, trabajador)
Responsabilidades: crear un objeto venta y lo asigna al historial de ventas
Precondiciones: existe relación cliente-tratamiento
Post-condiciones: Se crea objeto venta_total de tipo venta.
Se instancia venta_total con los datos de cliente, tratamiento y trabajador en el historial de ventas.
El atributo total de la clase venta se modifica con respecto a los atributos “precio” de la clase tratamiento y “descuento” de la clase cliente

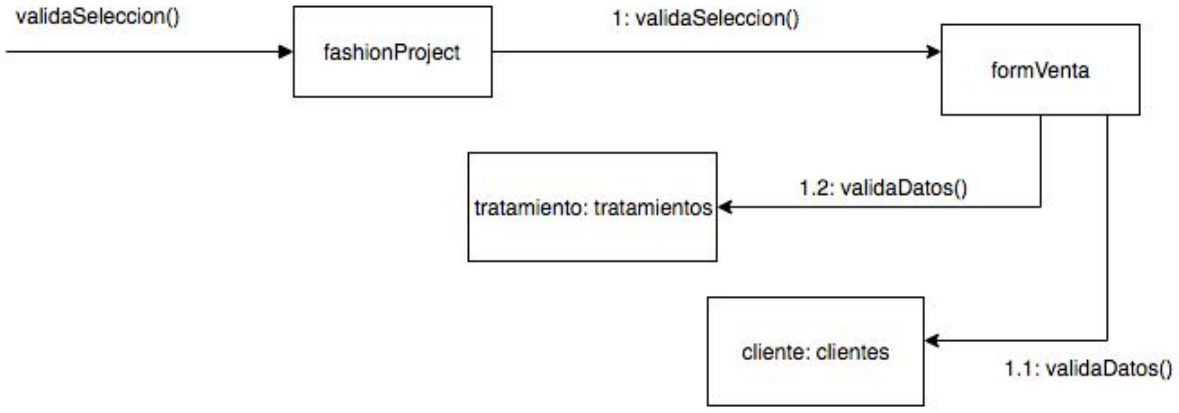
Diagramas de Colaboración:

Registrar Venta (Cliente ya registrado en el sistema)

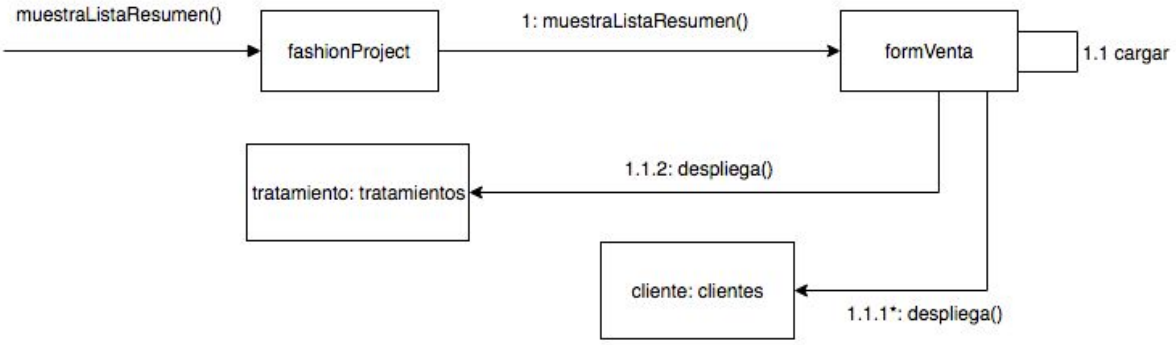
mostrarListaTratamientoListaCliente



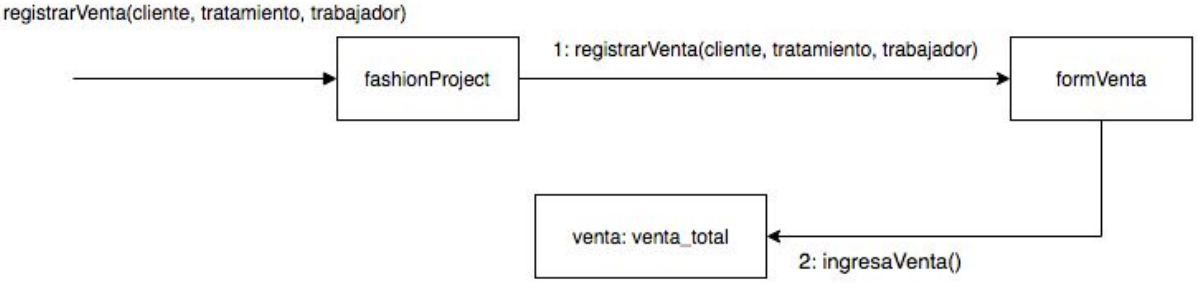
validaSeleccion



muestraListaResumen

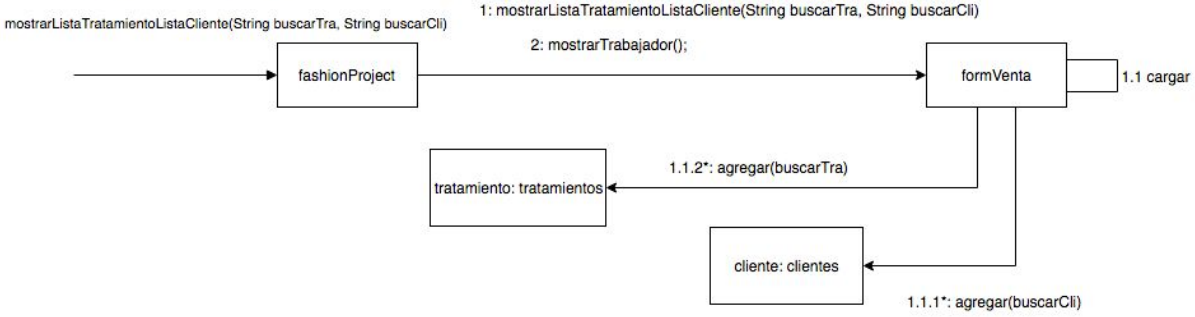


registrarVenta

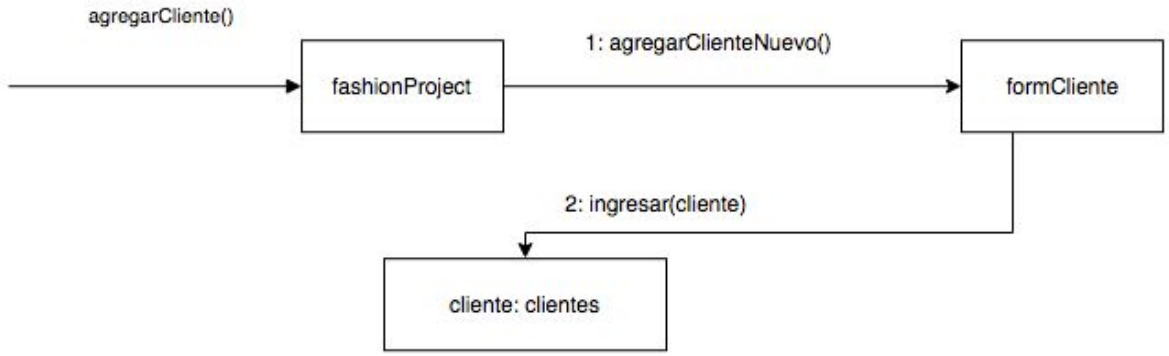


Registrar Venta (Cliente no está registrado en el sistema y agregar nuevo cliente)

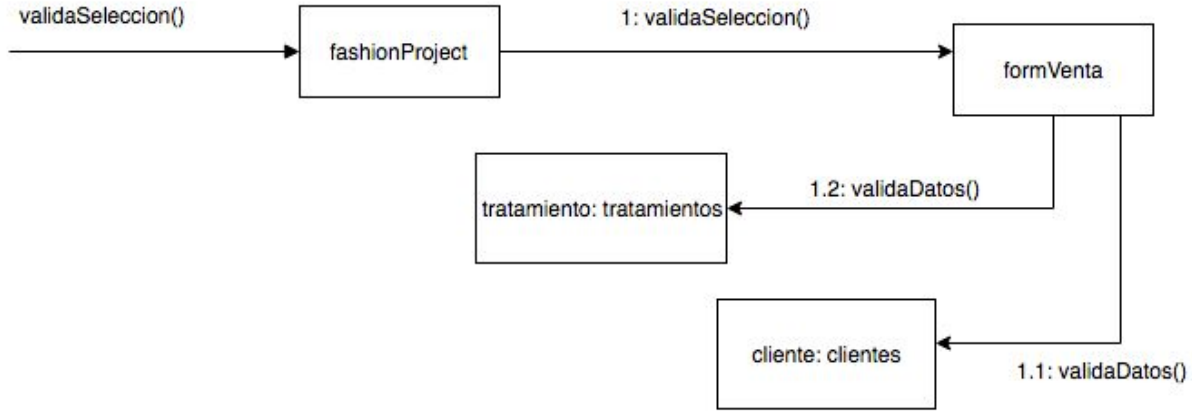
mostrarListaTratamientoListaCliente



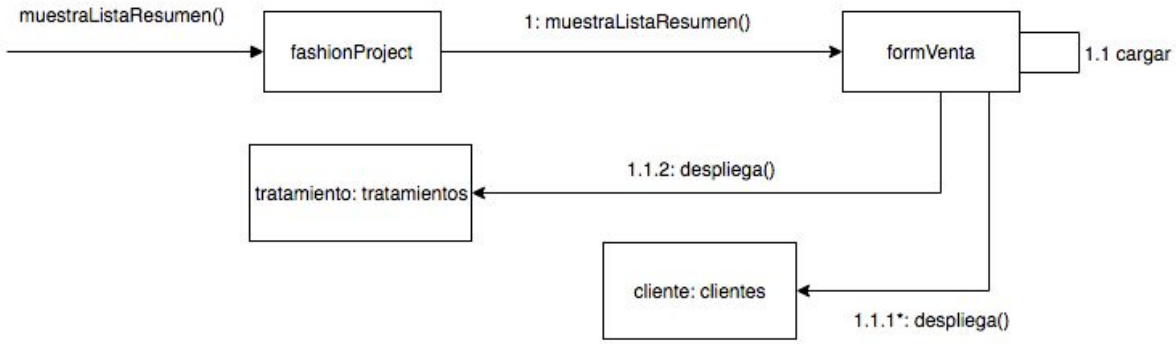
agregarCliente



validaSeleccion



muestraListaResumen



registrarVenta

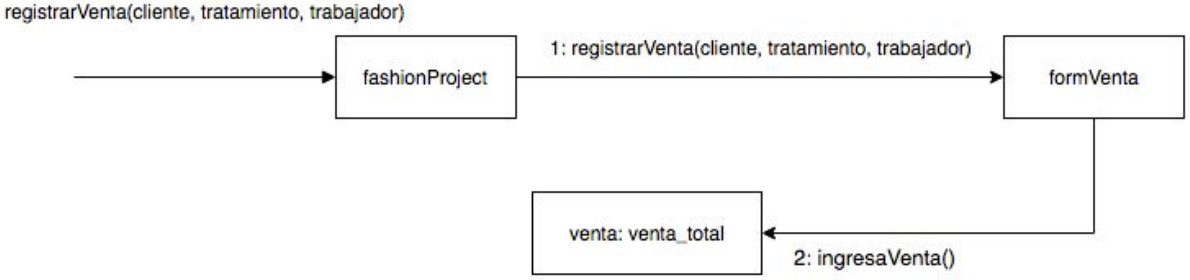


Diagrama de Estados:

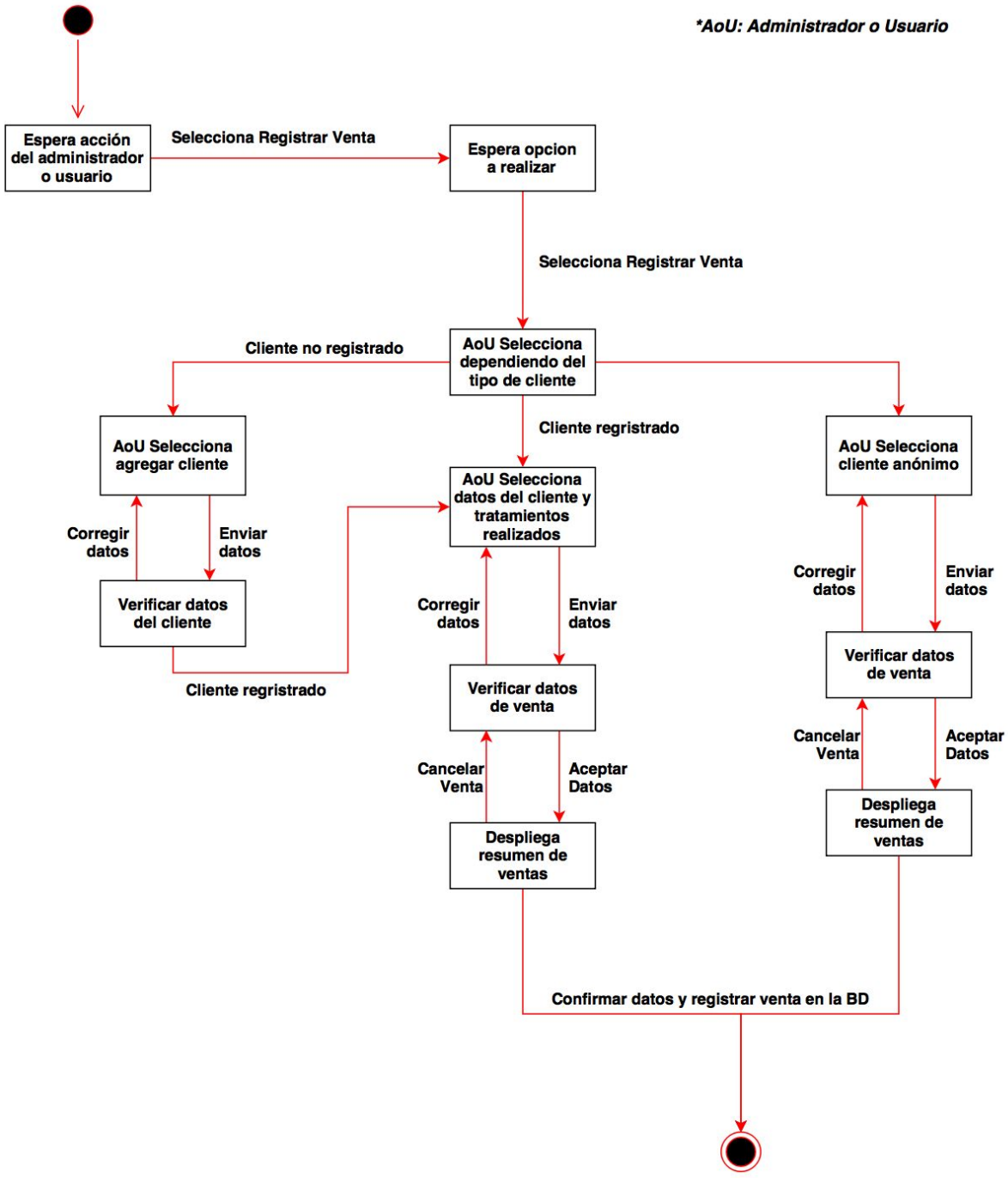
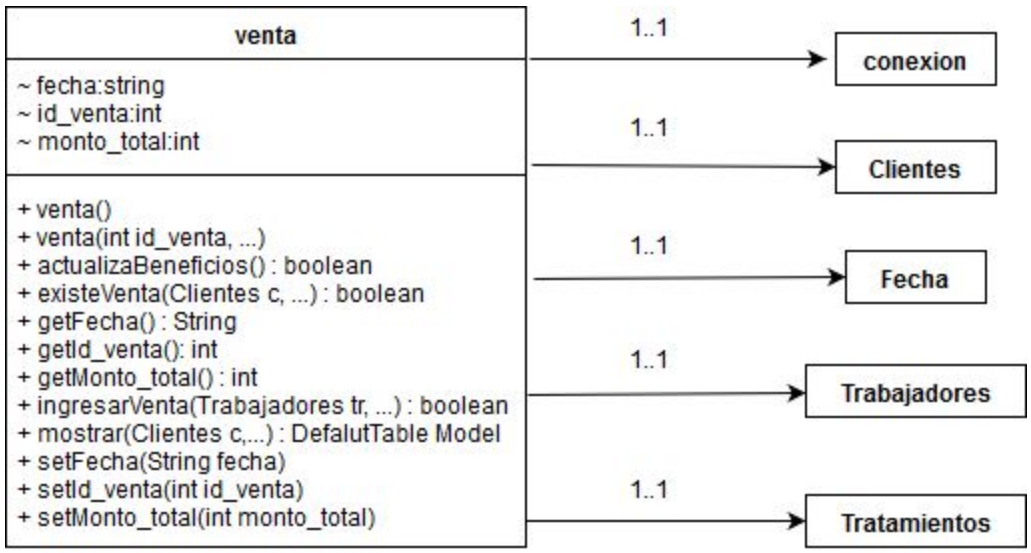
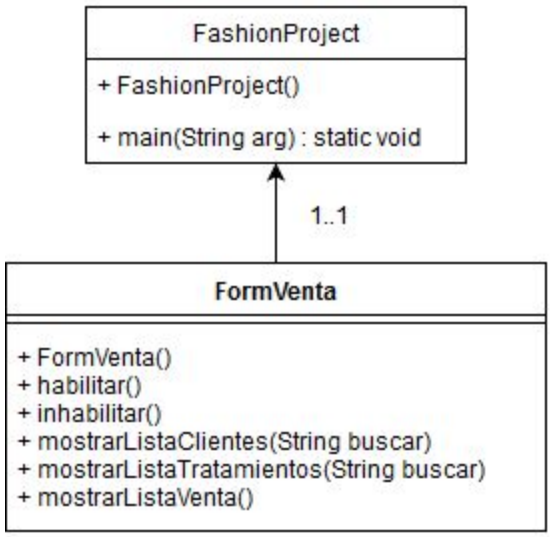


Diagrama de Clases:





Incremento 5:

Caso de uso: Gestión Gastos.

Actor: Usuario administrador.
Precondición: El usuario administrador debe estar registrado en el sistema y autorizado por el sistema.
OBS: para los cursos alternativos de modificar gastos, éstos deberán estar registrado en la base de datos.
Resumen: El usuario ingresa al sistema y gestiona los gastos, es decir, puede ingresar un nuevos gastos o también puede modificarlos. Los gastos a ingresar son Luz, Agua, Internet, Arriendo y otros.
Tipo: Primario - esencial

Curso normal de los eventos: Ingresar gasto.

Actor	Sistema
1) Usuario ingresa a la gestión de gastos.	2) Sistema despliega la ventana de gestión de gastos.
3) Usuario ingresa el valor de los gastos.	4) Sistema verifica los datos ingresados y despliega un mensaje de verificación.
6) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: El gasto ya se encuentra registrado.

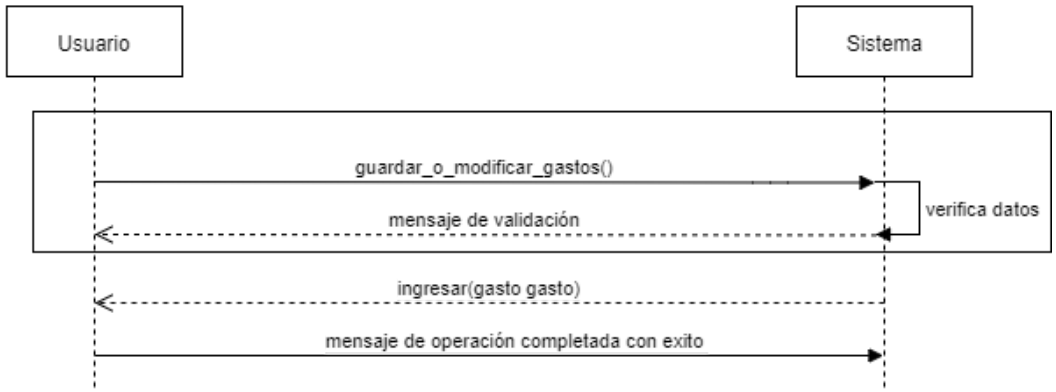
Actor	Sistema
3*) Usuario sobreescribe el nuevo valor de los gastos.	4) Sistema verifica los datos ingresados y despliega un mensaje de verificación.
6) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: Modificar Gasto.

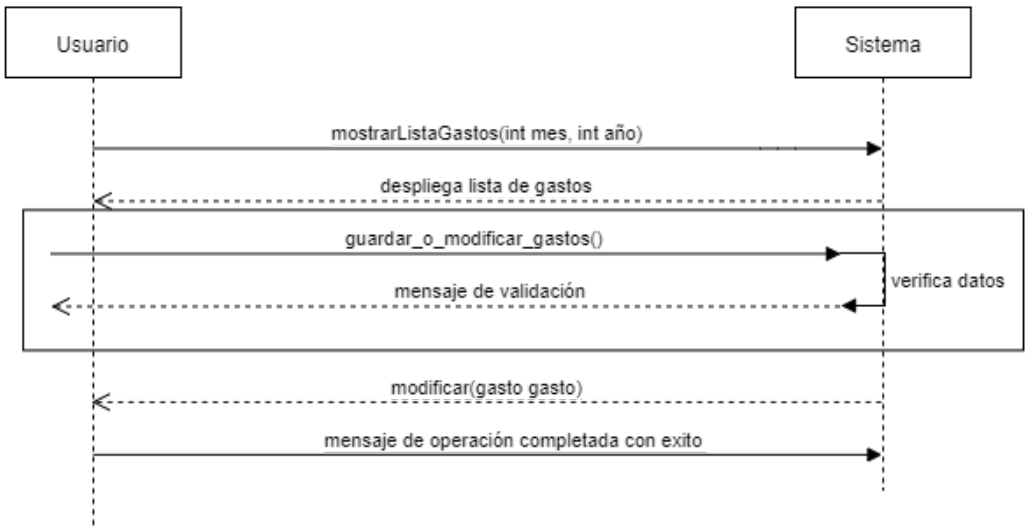
Actor	Sistema
3*) Usuario administrador busca un gasto al cual modificar su valor por mes y año.	4) Muestra lista de gastos que coincide con la búsqueda y habilita las casillas.
5) Usuario administrador ingresa los nuevos datos.	6) Se modifican los datos, desplegando un mensaje.
7) El usuario administrador realiza otra acción o sale del sistema.	

Diagramas de secuencia

Ingresar Gastos



Modificar Gastos



Contratos:

- Ingresar Gasto

Nombre: guardar_o_modificar_gastos()
Responsabilidades: ingresa los datos de los gastos a la base de datos
Precondiciones:
Post-condiciones: se crea el objeto gasto
se instancia un gasto de la clase gastos

Nombre: ingresar(gasto gasto)
Responsabilidades: Ingresa los valores de gasto a la base de datos.
Precondiciones: Existe la instancia de gasto
Post-condiciones: Se agregan los datos a la base de datos

- Modificar Gasto

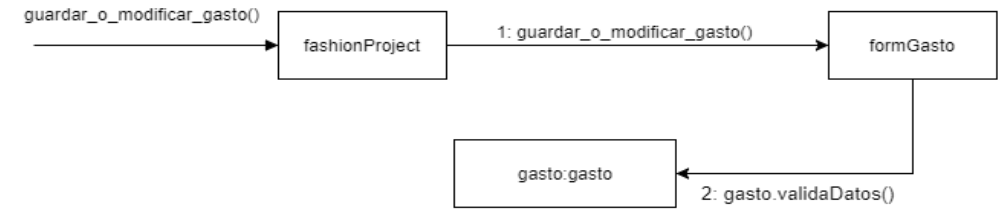
Nombre: mostrarListaGasto(int mes, int año)
Responsabilidades: mostrar lista con los gastos ingresados
Precondiciones: deben existir datos ingresados
Post-condiciones: se busca la lista de los objetos gastos y se cargan los datos preexistentes en las casillas habilitadas.

Nombre: guardar_o_modificar_gastos()
Responsabilidades: verifica y valida los datos ingresados por el usuario
Precondiciones:
Post-condiciones: se crea el objeto gasto
se instancia un gasto de la clase gastos

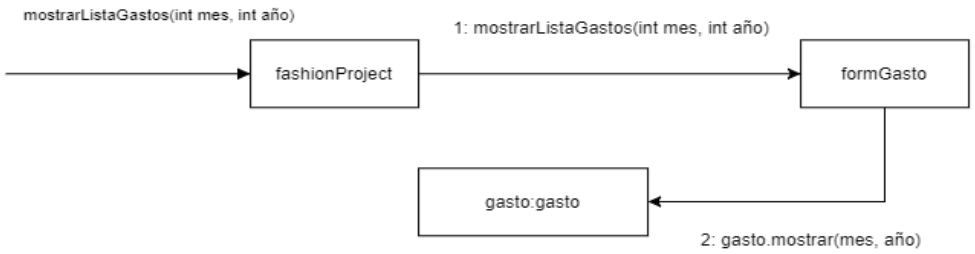
Nombre: modificar(gasto)
Responsabilidades: Actualiza los datos del gasto en la base de datos.
Precondiciones: Existe la instancia de gasto.
Post-condiciones: Se actualizan los datos en la base de datos

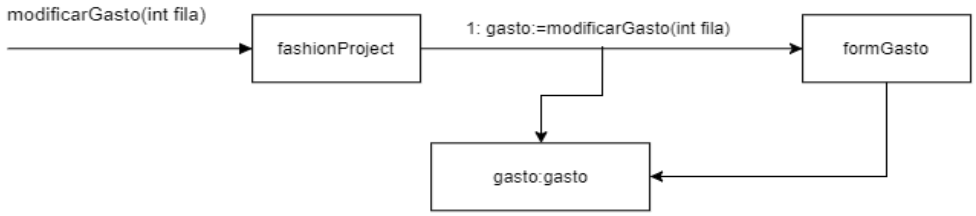
Diagramas de Colaboración:

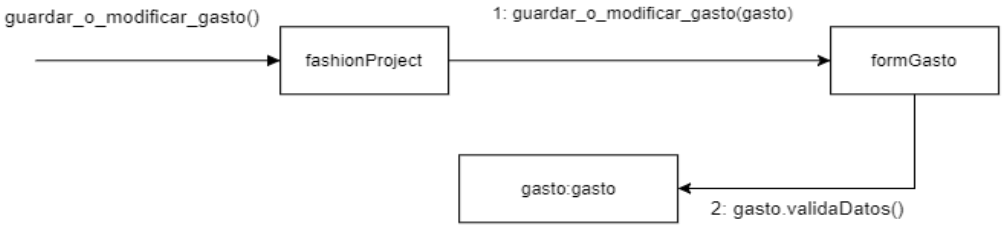
Ingresar Gastos



Modificar Gastos







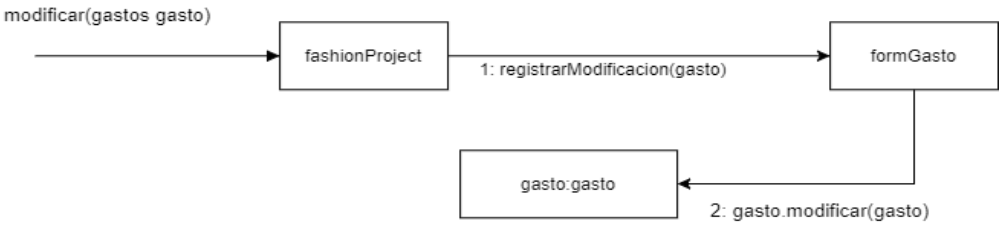


Diagrama de Estados:

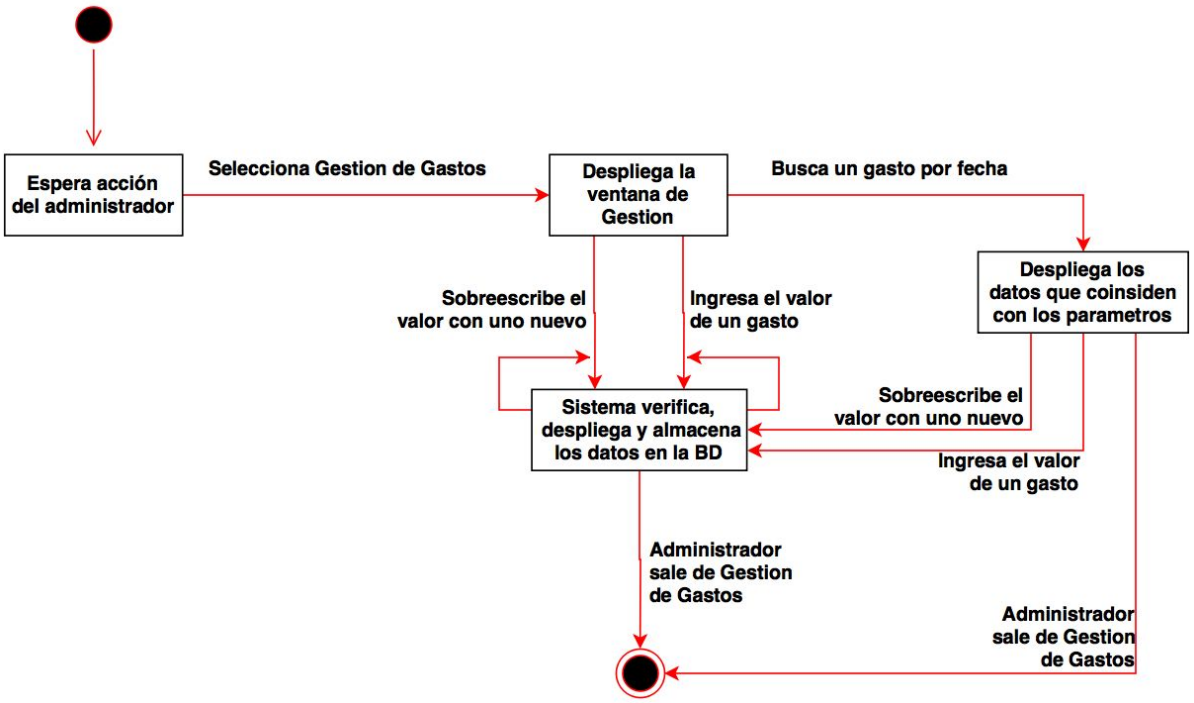
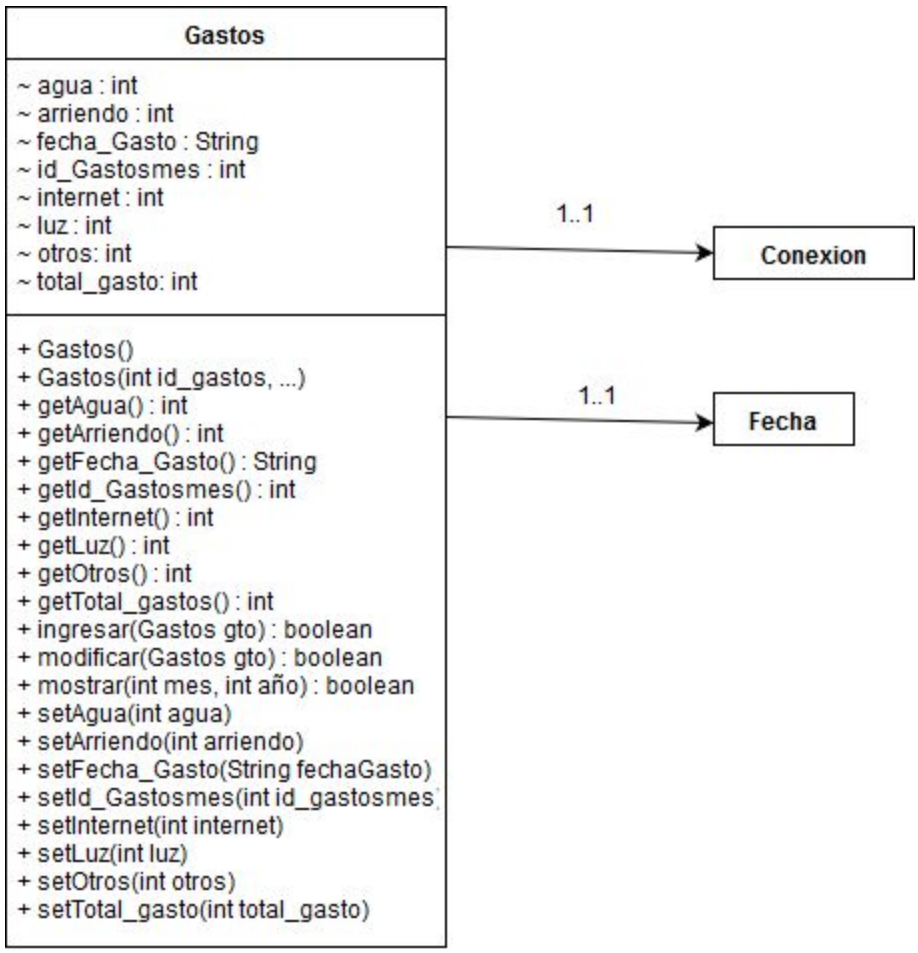
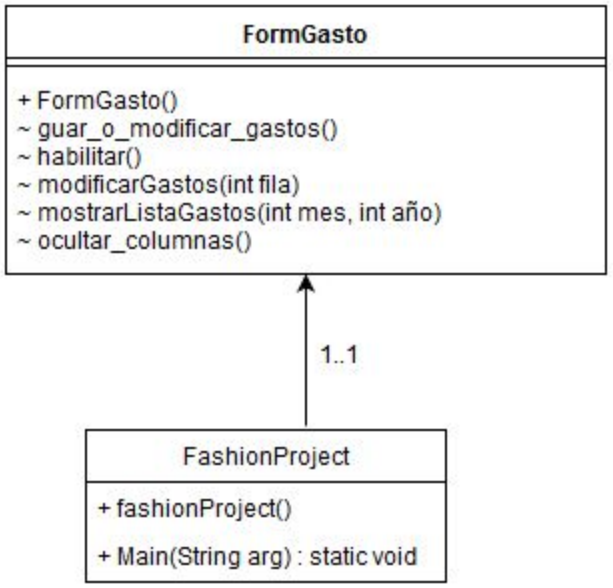


Diagrama de Clases:





Incremento 6:

Caso de uso: Generar Informes.

Actor: Usuario administrador.
Precondición: El usuario administrador debe estar registrado en el sistema y autorizado por el sistema.
Resumen: El usuario ingresa al sistema y selecciona la opción de generar informes, es decir, puede visualizar la información primordial de la empresa como pagos de los trabajadores, flujos de caja por mes y año.
Tipo: Primario - esencial

Curso normal de los eventos: Generar informe de pago semanal por cada trabajador.

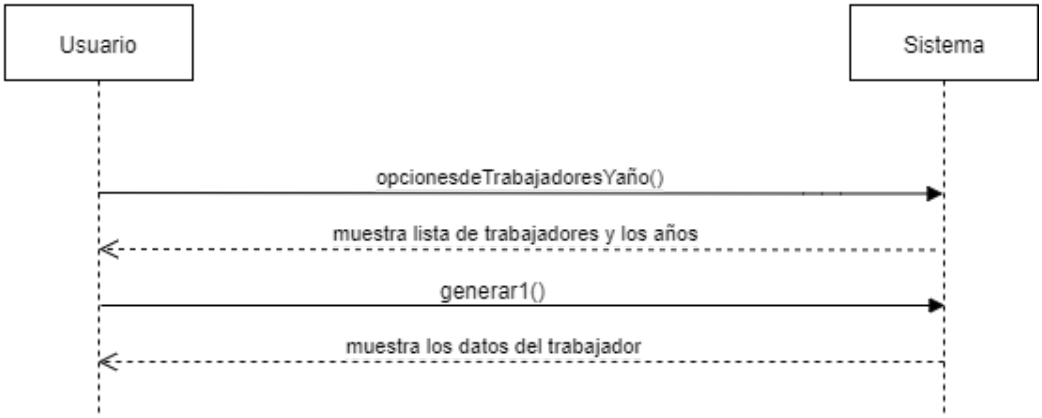
Actor	Sistema
1) Usuario ingresa a la sección de generar informes.	2) Sistema despliega la ventana de generación de informes con las opciones que puede elegir.
3) Usuario selecciona la opción de generar informes de pagos semanales por cada trabajador.	4) Sistema despliega la información asociada a cada trabajador.
5) El usuario administrador realiza otra acción o sale del sistema.	

Curso alternativo: Generar informe de flujos de caja por mes y año o solo por año.

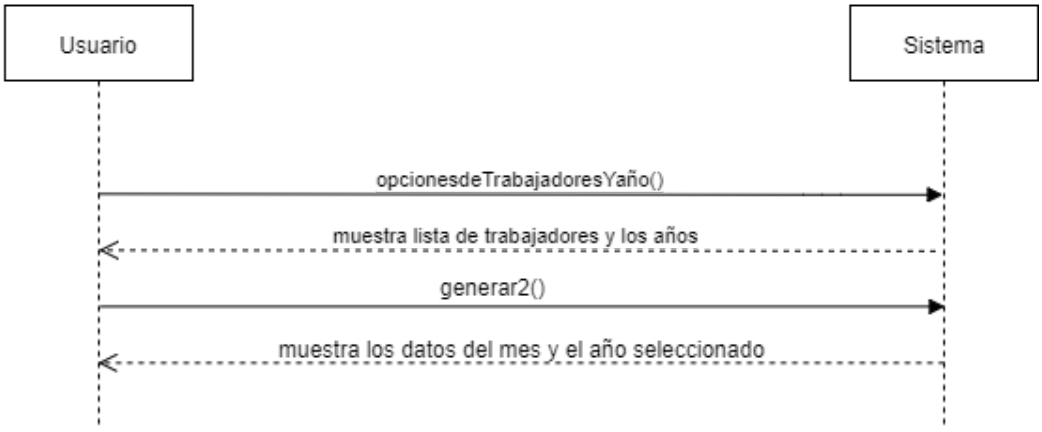
Actor	Sistema
3*) Usuario selecciona la opción de generar informes para ver los flujos de caja por mes y por año o solo por año	4) Sistema despliega la información asociada a cada mes y año seleccionado o año seleccionado.
5) El usuario administrador realiza otra acción o sale del sistema.	

Diagramas de Secuencia:

Generar informe de pago semanal por cada trabajador



Generar informe de flujos de caja por mes y año o solo por año



Contratos:

- Generar informe de pago semanal por cada trabajador

Nombre: opcionesdeTrabajadoresYaño()
Responsabilidades: mostrar los objetos de trabajadores y los elementos años de la fecha
Precondiciones: deben existir objetos de trabajadores y elementos años de la clase fecha

Post-condiciones: se muestran los objetos de trabajadores
se muestran los elementos de años

Nombre: generar1()
Responsabilidades: mostrar los datos del objeto seleccionado
Precondiciones: el objeto trabajador debe tener de ventas realizadas
Post-condiciones: se muestran los datos relacionados con el objeto trabajador

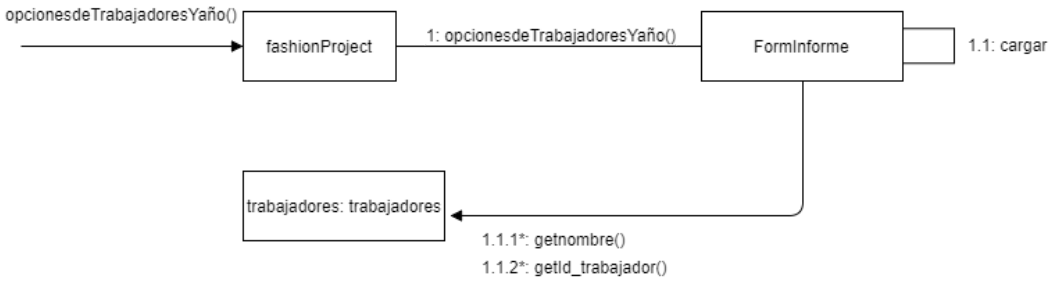
- Generar informe de flujos de caja por mes y año o solo por año.

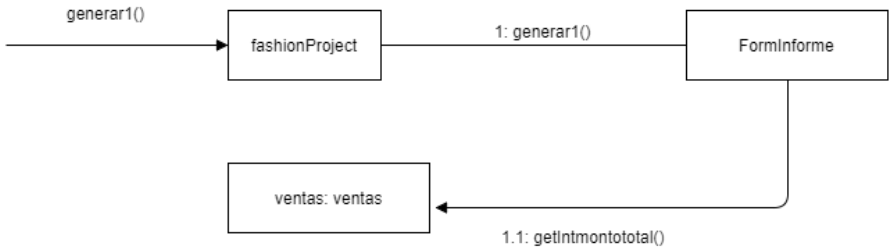
Nombre: opcionesdeTrabajadoresYaño()
Responsabilidades: mostrar los objetos de trabajadores y los elementos años de la fecha
Precondiciones: deben existir objetos de trabajadores y elementos años de la clase fecha
Post-condiciones: se muestran los objetos de trabajadores
se muestran los elementos años de la clase fecha

Nombre: generar2()
Responsabilidades: mostrar los datos del elemento seleccionado
Precondiciones: deben existir objetos ventas para el elemento año y mes
Post-condiciones: se muestran los datos relacionados con el elemento año

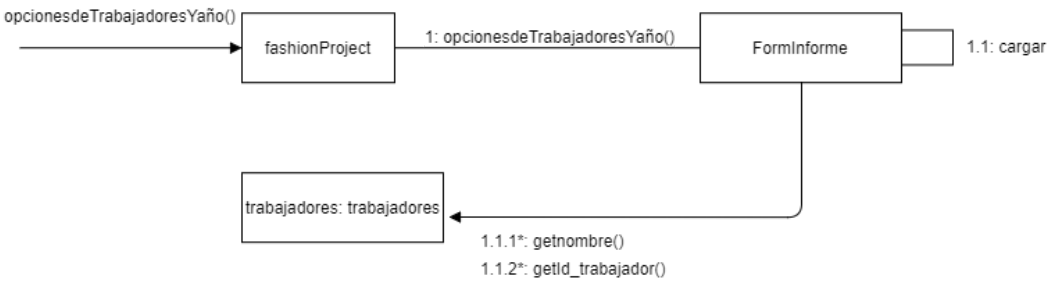
Diagramas de Colaboración:

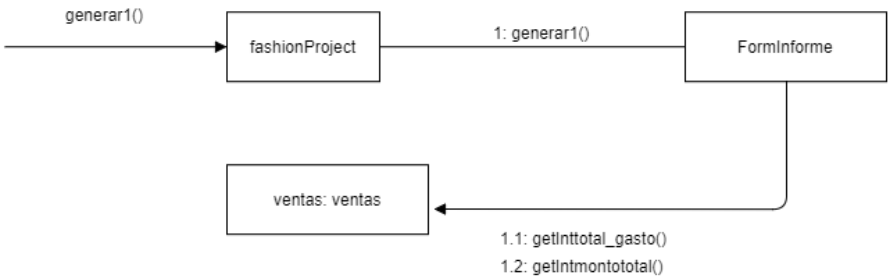
- **Generar informe de pago semanal por cada trabajador**





- **Generar informe de flujos de caja por mes y año o solo por año**





Diagramas de Estado:

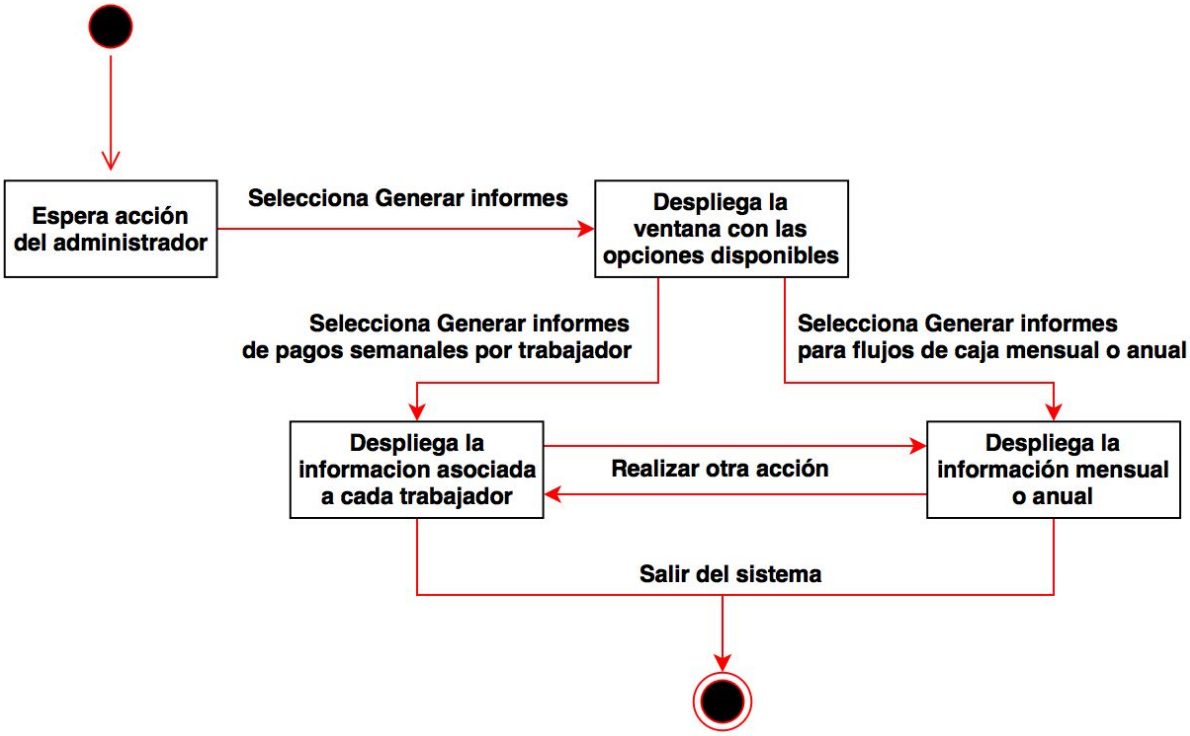
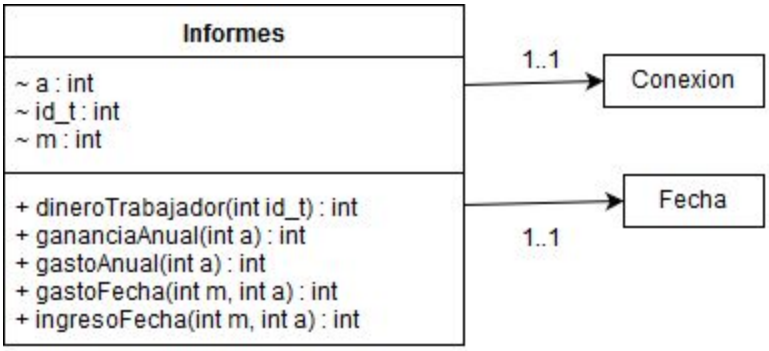
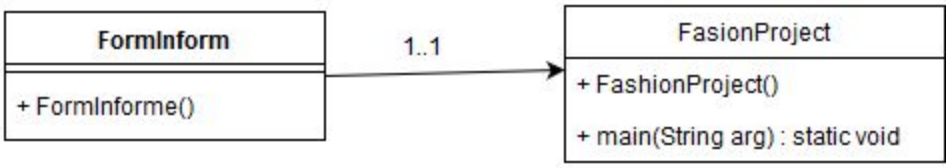


Diagrama de Clases:





Incremento 7:

Caso de uso: **Login.**

Actor: Usuario administrador, usuario normal.
Precondición: El usuario debe estar registrado en el sistema.
Resumen: El usuario ingresa los datos al sistema (nombre de usuario y clave) y este permite el ingreso con los privilegios correspondientes según sea el caso del tipo de usuario.
Tipo: Primario - esencial

Curso normal de los eventos: Ingresar al sistema.

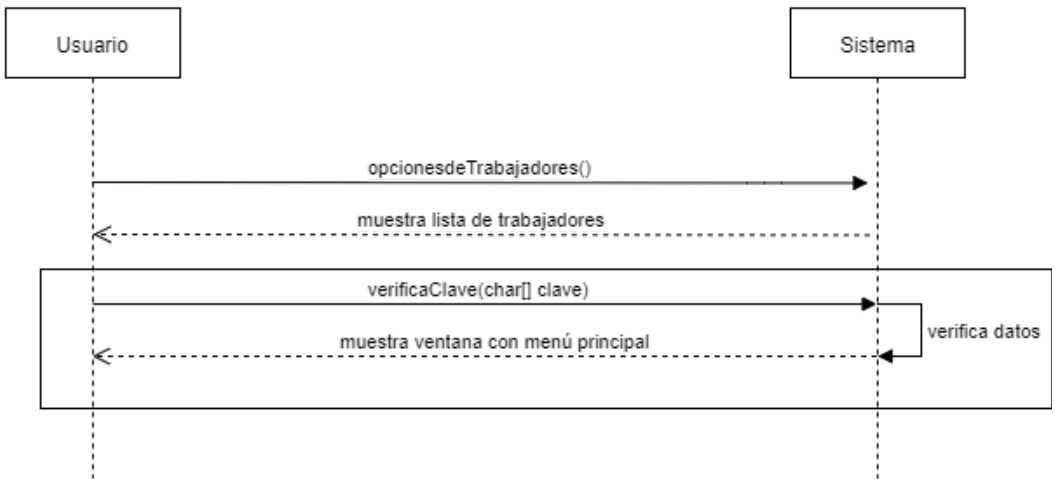
Actor	Sistema
1) Usuario ingresa su nombre de usuario y su clave.	2) Sistema verifica los datos ingresados y despliega la ventana del menú con los botones habilitados.
3) El usuario realiza un acción o sale del sistema.	

Curso alternativo: Datos erróneos.

Actor	Sistema
*1) Usuario ingresa su nombre de usuario y su clave con errores.	2) Sistema verifica los datos ingresados y despliega un mensaje de error, el sistema limpia los campos.

Diagramas de Secuencia:

Ingresar al sistema



Contratos:

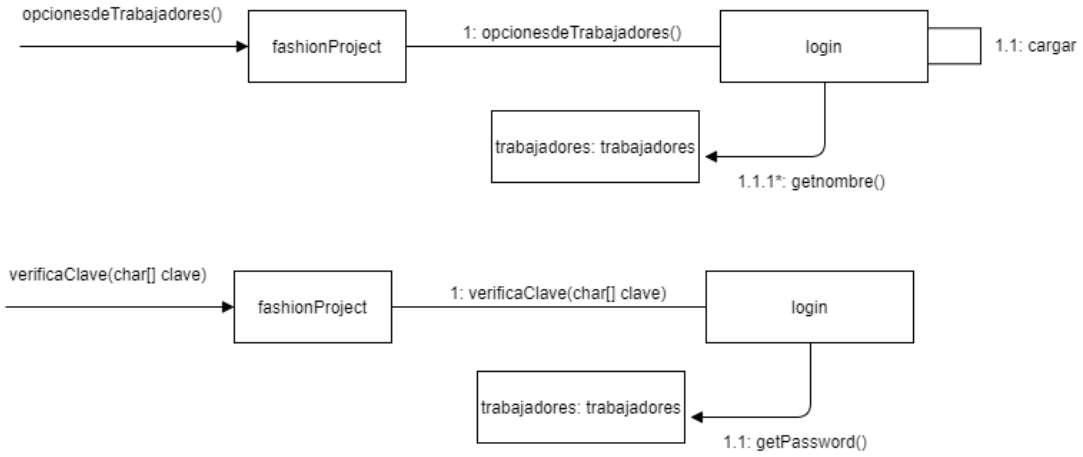
- Ingresar al sistema

Nombre: opcionesdeTrabajadores()
Responsabilidades: mostrar la lista de los trabajadores
Precondiciones: debe existir una lista con objetos trabajadores registrados en el sistema
Post-condiciones: muestra la lista de objetos en un panel

Nombre: verificaClave(char[] clave)
Responsabilidades: validar que la clave coincida con el objeto trabajador seleccionado
Precondiciones: debe existir una clave para cada objeto trabajador
Post-condiciones: se asigna un objeto trabajador de la lista a la sesión del sistema

Diagramas de Colaboración:

Ingresar al sistema



Diagramas de Estado: