ENTREGA SUBEQUIPO DE DISEÑO – ANÁLISIS

Moisés Gautier Gómez
Francisco Javier Gómez del Olmo
Julio Ros Martínez

20 DE MARZO DE 2013

Universidad de Granada

Contenido

Control de Versiones

- 1º Control de versión

Fecha	Versión	Descripción
20/3/2013	1.1	Entrega diseño 20-3-2013

Análisis

Identificar clases, atributos y relaciones

En este punto se buscan las abstracciones más significativas que identifiquen los aspectos claves del dominio del problema. Las clases conceptuales deben interrelacionarse entre sí a través de las relaciones para satisfacer las necesidades de información o de comportamiento que demandan los casos de uso y así comprender mejor el modelo.

A continuación se detallan las clases, atributos y relaciones entre las clases que se han obtenido para esta primera iteración del problema planteado.

Clase: Fundación

Atributos:

- ciudad (String): Nombre la ciudad o localidad de la fundación de baloncesto.
- codPostal (int): Código postal de la ciudad o localidad de la fundación de baloncesto.
- direccion (String): Dirección postal de la ciudad o localidad de la fundación de baloncesto.
- email (String): Correo eléctronico de la fundación.
- nombre (String): Nombre completo de la fundación de baloncesto.
- numeroCuenta (String): Número de cuenta bancario de la fundación donde se realizaran los ingresos de las mensualidades.
- telefono (int): Número de teléfono de la fundación con el prefijo de la ciudad o localidad incluido.

Relaciones:

- Asociación con la clase Equipo.

Clase: Categoría

Atributos:

- descripción (String): Breve descripción de la categoría del alumno.
- edadMaxima (int): La edad máxima posible a la que se puede acceder a la categoría.
- idCategoria (int): Número identificador de la categoría.
- tipo (String): Nombre de la categoría del alumno.

Relaciones:

- Asociación con la clase Equipo.
- Asociación con la clase Grupo.

Clase: Equipo

Atributos:

- idEquipo (int): Número identificador del equipo.

Relaciones:

- Asociación con la clase Categoría.
- Asociación con la clase Entrenador.
- Asociación con la clase Fundación.
- Asociación con la clase Alumno.
- Asociación con la clase Temporada.

Clase: Temporada

Atributos:

- curso (string): año correspondiente al curo de la temporada (ejemplo: 2012-2013 sería 12/13).
- idTemporada (int): Número identificador de la temporada.

Relaciones:

- Asociación con la clase Equipo.
- Asociación con la clase Grupo.
- Asociación con la clase Alumno.
- Asociación con la clase Temporada.

Clase: Rango (clase de asociación entre las clases Entrenador y Equipo)

Atributos:

- idRango (int): Número identificador del rango
- tipo (TipoEntrenador): Campo que describe que tipo de entrenador es para el equipo, si el primero o el segundo, mediante el tipo enumerado TipoEntrenador.

Relaciones:

- Clase de asociación entre Entrenador y Equipo.

Clase: Pago Temporada (clase de asociación entre las clases Temporada y Alumno)

Atributos:

- idPagoTemporada (int): Número identificador del pago por temporada.

Relaciones:

- Asociación con la clase Cuota precio (composición de esta última)

Clase: Cuota Precio

Atributos:

- idCuotaPrecio (int): Número identificador del precio de la cuota.
- importe (float): Valor numérico del importe de la mensualidad del alumno.
- pagado (boolean): Valor booleano que establece si la mensualidad ha sido pagada o no por el alumno. Se modificará una vez se haya recibido del banco el documento con los pagos del mes.
- fechaPago (date): Fecha en la que se realizó el pago.

Relaciones:

- Asociación con la clase Pago Temporada.
- Asociación con la clase Pago Actividad.

Clase: Alumno

Atributos:

- codigoPostal (int): Código postal de la ciudad o localidad donde resida el alumno.
- colegio (string): Colegio al que está inscrito durante el curso.
- domicilio (string): Dirección postal donde vive el alumno.
- email (string): Correo electrónico del alumno o del tutor para recibir notificaciones pertinentes de la fundación o algún otro relacionado.
- fechaNacimiento (date): Fecha de nacimiento del alumno para establecer la categoría a la que pertenece.
- idAlumno (int): Número identificador del alumno.
- localidad (string): Nombre de la ciudad o localidad donde resida el alumno.
- nombre (string): Nombre del alumno.
- nombreMadre (string): Nombre de la madre del alumno (no tiene por qué llevar asociado el apellido ya que se podría extraer de los apellidos del alumno).
- nombrePadre (string): Nombre del padre del alumno (no tiene por qué llevar asociado el apellido ya que se podría extraer de los apellidos del alumno).
- numeroCuenta (string): Número de cuenta bancario donde el alumno domiciliará las mensualidades de la fundación de baloncesto.
- observaciones (string): Campo dedicado a datos que los entrenadores del alumno quieran anotar de su rendimiento o como campo para anotaciones de cualquier índole.
- primerApellido (string): Primer apellido del alumno.

- provincia (string): Provincia a la que pertenece la ciudad o localidad del alumno durante el curso.
  segundoApellido (string): Segundo apellido del alumno.
- talla (TallaAlumno): Este campo corresponde con la talla del alumno que pertenece al tipo enumerado TallaAlumno en donde se establecen todos los valores posibles que puede tomar el campo.
- telFijo (int): Número de teléfono fijo del alumno/tutor durante el curso.
- telMovil (int): Número de teléfono móvil del alumno/tutor durante el curso.

Relaciones:

- Asociación con la clase Temporada.
- Asociación con la clase Grupo.
- Asociación con la clase Actividad.

Clase: Entrenador (clase de especialización de Usuario)

Atributos:

Relaciones:

- Asociación con la clase Equipo.
- Asociación con la clase Grupo.

Clase: Grupo Entrenamiento

Atributos:

- horarios (date): Horarios del grupo de entrenamiento en donde está inscrito el alumno durante el curso. El dato contendrá los días de entrenamiento para la categoría a la que pertenezca, así como las horas disponibles por cada dupla de días.
- idGrupo (int): Número identificador del alumno.

Relaciones:

- Asociación con la clase Entrenador.
- Asociación con la clase Categoría.
- Asociación con la clase Alumno.
- Asociación con la clase Temporada.

Clase: Usuario

Atributos:

- clave (string): Clave del usuario para acceder a la aplicación. En principio su almacenamiento podrá seguir algún tipo de cifrado seguro como MDA5.
- dni (string): DNI del usuario.
- idUsuario (int): Número identificador del usuario
- nombre (string): Nombre del usuario.

- primerApellido (string): Primer apellido del usuario.
- segundoApellido (string): Segundo apellido del usuario.

Relaciones:

Clase: Pago Actividad (clase de asociación entre Alumno y Actividad)

Atributos:

- idPagoActividad (int): Número identificador del pago de la actividad.
- recibo (string): Ruta relativa o completa del archivo que contiene la imagen del recibo/pago/domiciliación del pago de la actividad.

Relaciones:

- Asociación con la clase Cuota precio (composición).

Clase: Actividad

Atributos:

- descripcion (string): Descripción breve de la actividad para consultas por el sistema.
- fechaFin (date): Fecha fin de la actividad para la temporada.
- fechaInicio (date): Fecha inicio de la actividad para la temporada.
- idActividad (int): Número identificador de la actividad.

Relaciones:

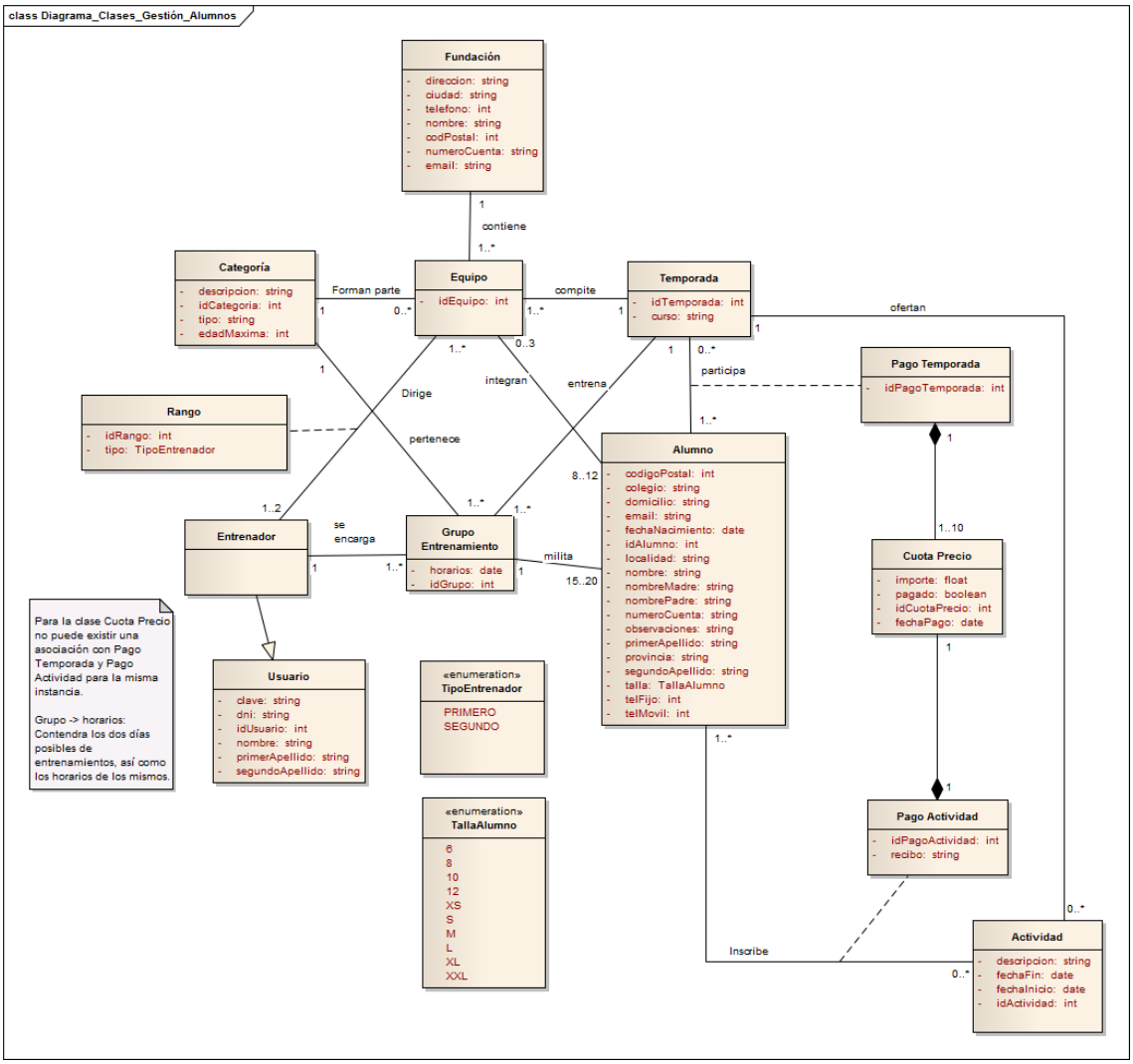
- Asociación con la clase Alumno.
- Asociación con la clase Temporada.

Modelado estático – Diagrama de clases

El modelo estático proporciona mecanismos para describir y representar las interrelaciones estructurales que se establecen entre las clases conceptuales de un modo visual y compacto.

- Proporciona una estructura estática de las clases conceptuales.
- Al ser un modelo de análisis sólo muestra las clases conceptuales extraídas del dominio de aplicación (dominio del problema) y sus relaciones, no componentes software pertenecientes a la solución.
- La reducida barrera que propugna la orientación a objetos entre el problema y la solución, posibilita que el modelo estático inspire la construcción del modelo de diseño.
- Definen un vocabulario común entre clientes y desarrolladores.
- Una vez que el modelo es estable, la descripción de cada clase será tan detallada como sea posible.
- Elimina la vaguedad en la definición de las clases del dominio del problema.

A continuación se describe el diagrama de clases obtenido para el problema especificado.



Modelado del comportamiento externo – Contratos

Describen el comportamiento detallado del sistema en función de los cambios de estado de los objetos cuando se invoca una operación del sistema.

Es un documento que describe lo que una operación se propone lograr, sin decir cómo se conseguirá.

- Define la especificación de una operación sin entrar en su implementación.
- Suele redactarse con un estilo declarativo.

A continuación se describen los contratos de las operaciones de tres de los casos de uso: Dar de alta un usuario, Modificar un alumno e Introducir Pago.

Contratos de Caso de uso: Dar de alta un usuario

Nombre	CrearNuevoUsuario (DNI:string, Nombre:string, Apellidos:string, Clave:string)
Responsabilidades	Introducir un usuario en el sistema
Tipo	Sistema
Referencias cruzadas	Caso de uso “Dar de Alta Usuario”
Notas	-
Excepciones	-
Salida	-
Precondición	No existe ninguna entrada en el sistema del nuevo usuario.
Postcondición	Se crea un nuevo usuario (creación de instancia).

Nombre	comprobarUsuario (DNI:string)
Responsabilidades	Comprueba si existe o no en el sistema una entrada para el DNI pasado como parámetro a la función.
Tipo	Sistema
Referencias cruzadas	Caso de uso “Dar de Alta Usuario”
Notas	Se hace una consulta al sistema con la información del DNI del usuario (consulta sobre la base de datos). Si se encuentra una coincidencia con el DNI pasado como parámetro se devolverá true o false a la variable validar (comprobar validez del dato).
Excepciones	-
Salida	-
Precondición	Se ha creado una nueva instancia de la clase Usuario.
Postcondición	-

Nombre	generarError()
Responsabilidades	Genera un mensaje de error si el usuario comprobado no existe.
Tipo	Sistema
Referencias cruzadas	Caso de uso “Dar de Alta usuario”
Notas	La operación CreaUsuario(…) crea una instancia temporal, la cual se comprueba si ya ha sido creada anteriormente. Si ha sido creada, se manda un mensaje de error y se elimina esa instancia temporal.
Excepciones	-
Salida	-
Precondición	El usuario debe estar registrado
Postcondición	-

Nombre	ActualizarUsuario (DNI:string, nombre:string, apellidos:string, clave:string)
Responsabilidades	Actualizar la información referente a un usuario
Tipo	Sistema
Referencias cruzadas	Caso de uso “Dar de Alta Usuario”
Notas	-
Excepciones	-
Salida	-
Precondición	El usuario no debe estar registrado.
Postcondición	Guarda la información referente a un usuario.

Contratos de Caso de uso: Consultar Alumno

Nombre	consultarAlumno()
Responsabilidades	Consultar los datos de un alumno existente en el sistema y opcionalmente poder modificarlos.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”
Notas	El sistema muestra las dos formas posibles de elegir un alumno: por su ID o mediante una lista de alumnos.
Excepciones	-
Salida	-
Precondición	Que haya al menos un alumno dado de alta en el sistema.
Postcondición	-

Nombre	elegirOpcion(opción:Enumerado)
Responsabilidades	Elegir entre las dos opciones de visualizar el alumno que se va a modificar, bien por su ID o bien con una lista de alumnos.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”
Notas	Si se elige la opción de visualizar un alumno mediante una lista muestra la lista de alumnos y si se elige la opción de buscar al alumno por su ID el usuario tendrá que introducir el ID del alumno.
Excepciones	-
Salida	-
Precondición	-
Postcondición	-

Nombre	seleccionarAlumnoLista()
Responsabilidades	Seleccionar un alumno que va a ser consultado de una lista de alumnos existentes en el sistema.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”.
Notas	-
Excepciones	-
Salida	datosAlumno.
Precondición	-
Postcondición	-

Nombre	seleccionarAlumnoID(idAlumno:int)
Responsabilidades	Seleccionar un alumno que va a ser consultado mediante su idAlumno.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”.
Notas	-
Excepciones	Si el idAlumno es erróneo, indicar que se cometió un error.
Salida	datosAlumno.
Precondición	Que el ID del alumno introducido exista en el sistema.
Postcondición	-

Nombre	modificarDatos(datos)
Responsabilidades	Modificar los datos referentes a un alumno.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”.
Notas	datos={codigoPostal, colegio, domicilio, e-mail, fechaNacimiento,localidad, nombrePadre, nombreMadre, numeroCuenta, observaciones, primerApellido, provincia, segundoApellido, talla, telFijo, telMovil}
Excepciones	-
Salida	-
Precondición	Que el alumno exista en el sistema.
Postcondición	-

Nombre	guardarCambios()
Responsabilidades	Guardar los cambios que se hayan producido en los datos del alumno.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”.
Notas	-
Excepciones	-
Salida	mensajeOK.
Precondición	Que todas las operaciones para modificar los datos del alumno se hayan efectuado correctamente.
Postcondición	Se modificaron los datos de un objeto “Alumno” que ya existía en el sistema.

Nombre	generarError()
Responsabilidades	Generar un mensaje de error debido a que los datos introducidos no sean correctos.
Tipo	Sistema.
Referencias cruzadas	Caso de Uso “Consultar alumno”.
Notas	-
Excepciones	-
Salida	mensajeError.
Precondición	-
Postcondición	-

Contratos de Caso de uso: Introducir Pago

Nombre	IntroducirDatosPago(cuentaBancaria:string, Importe:float, EstadoPagado:boolean, tipoPago:string)
Responsabilidades	Introducir los datos del pago en el sistema.
Tipo	Sistema.
Referencias cruzadas	Caso de uso “Introducir Pagos”.
Notas	-
Excepciones	Si los datos se introducen de forma incorrecta, informar que se cometió un error.
Salida	-
Precondición	-
Postcondición	El sistema obtuvo la información de un pago.

Nombre	AlmacenarPagoTemporada(cuentaBancaria:string, importe:float, EstadoPagado:boolean)
Responsabilidades	Almacenar en la BD el pago referente a una mensualidad.
Tipo	Sistema.
Referencias cruzadas	Caso de uso “Introducir Pagos”.
Notas	El alumno que paga se sabe a partir del número de la cuenta bancaria, y la temporada se sabe a partir de dicho alumno.
Excepciones	Si no se han podido almacenar los datos, indicar que se cometió un error.
Salida	-
Precondición	-
Postcondición	Se creó un pago Temporada (creación de instancia). Se asoció el pago Temporada a un alumno (asociación formada) Se asoció el pago Temporada a una temporada (asociación formada)

Nombre	AlmacenarPagoActividad(cuentaBancaria:string, importe:float, EstadoPagado:boolean)
Responsabilidades	Almacenar en la BD el pago referente a una actividad “act”.
Tipo	Sistema.
Referencias cruzadas	Caso de uso “Introducir Pagos”.
Notas	El alumno que paga se sabe a partir del número de la cuenta bancaria, y la actividad se sabe a partir de dicho alumno.
Excepciones	Si no se han podido almacenar los datos, indicar que se cometió un error.
Salida	-
Precondición	-
Postcondición	Se creó un pago Actividad (creación de instancia). Se asoció el pago Actividad a un alumno (asociación formada) Se asoció el pago Actividad a una temporada (asociación formada)

Modelado del comportamiento externo – Diagrama de secuencia de las operaciones

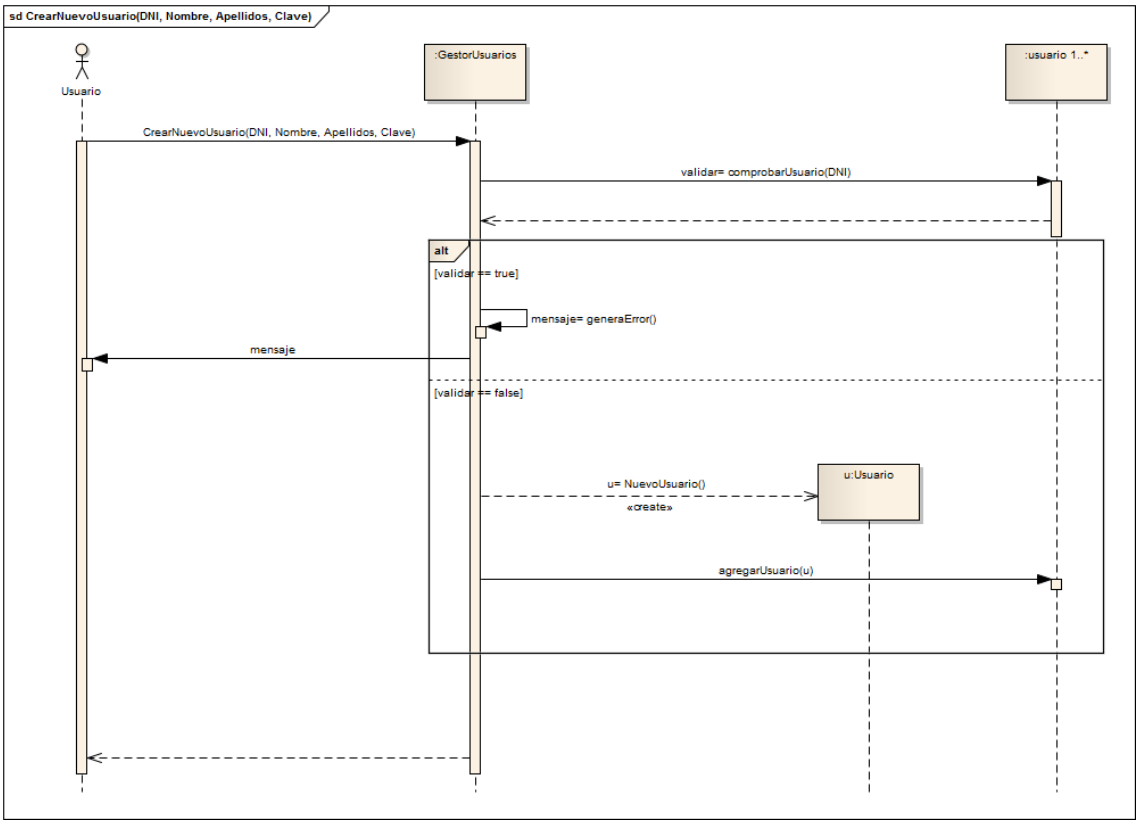
Es un documento que describe lo que una operación se propone lograr, sin decir cómo se conseguirá.

- Define la especificación de una operación sin entrar en su implementación.
- Suele redactarse con un estilo declarativo.

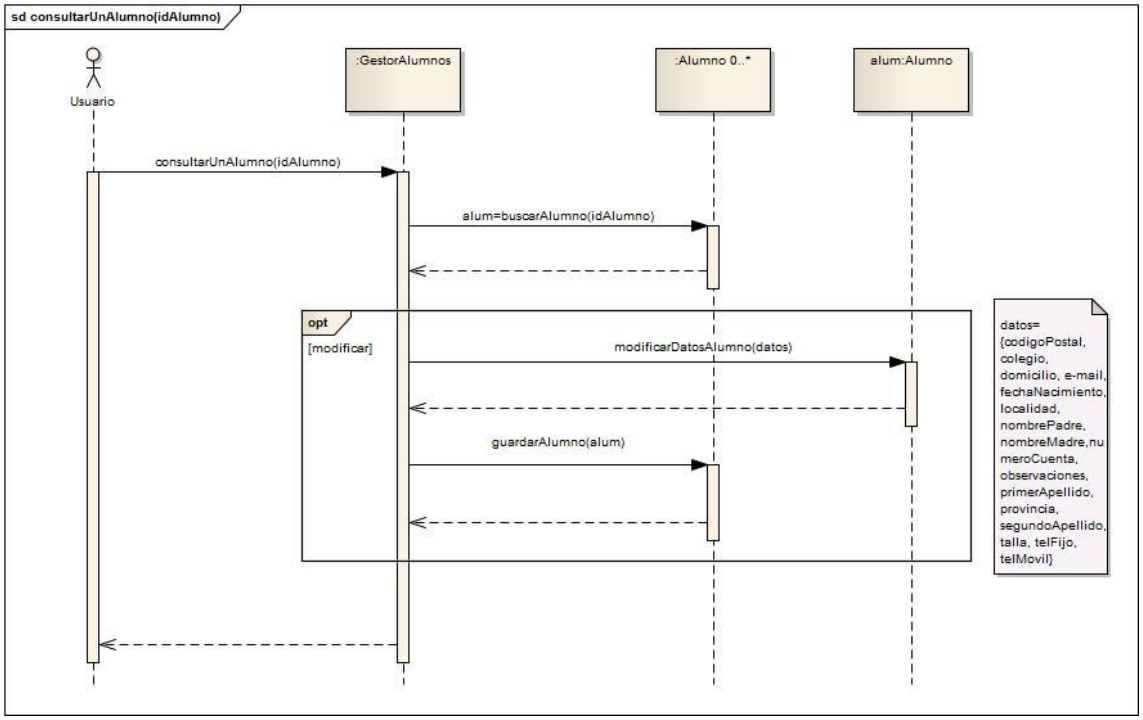
En este caso no serán diagramas de secuencia de operaciones del sistema como tales, sino de las operaciones que definen los contratos de los diagrama de secuencia obtenidos en la anterior entrega.

A continuación se describen los diagramas de secuencia de las operaciones de los contratos anteriores.

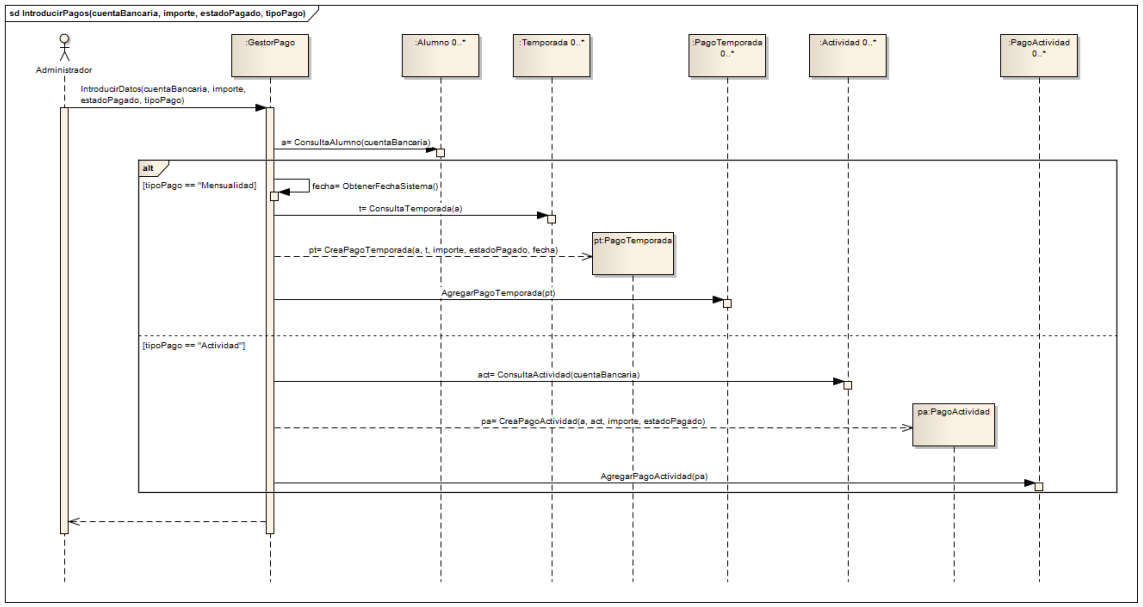
Dar de alta un alumno



Modificar Alumno



Introducir Pagos



Anexo control de versiones

Fecha: 20/3/2013 **Versión:** 1.1

- Identificar clases, atributos y relaciones v.1.1.
- Modelado estático – Diagrama de clases v.1.1.
- Modelado del comportamiento externo – Contratos v.1.1
- Modelado del comportamiento externo – Diagrama de secuencia de los contratos v1.1.