Índice de contenido

Objetivos....3
Objetivos Generales....3
Objetivos Específicos....3
REQUERIMIENTOS....4
Modelo de dominio....4
Modelo de negocio....5
REQUISITOS....5
Identificación de actores y casos de uso....5
Actores....5
Casos de Uso....5
Priorización de casos de uso....6
Detallar casos de uso....6
Caso de uso : Gestionar Estados....9
Caso de Uso : Generar Codigo....10
ANÁLISIS....11
Análisis de arquitectura....11
Análisis de Casos de Uso....12
Caso de Uso : Gestionar clases....12
Caso de Uso : Gestoinar relaciones....13
Caso de Uso : Gestionar estados....14
Caso de Uso : Generar codigo fuente....14
Análisis de Clases....15
Clases Interfaz....15
Clases de Control....16
Clases Entidad....17
Análisis de paquetes....19
Gestión de modelo....19
Gestión de estados....19
Generar codigo....20
DISEÑO....20
Diseño de Arquitectura....20
Diseño de Casos de Uso....21
Caso de Uso: Gestionar clases....21
Caso de Uso : Gestionar relaciones....22
Caso de Uso : Gestionar estados....23
Caso de Uso : Generar codigo....23
Diseño de Clases 25
Diseño de clases....25
IMPLEMENTACIÓN....26
Implementación de la Arquitectura....26
Implementación de Subsistemas....26
Gestión modelo....26
Gestión de estados....27
Gestión generación de codigo....27

Objetivos

Objetivos Generales

Desarrollar una herramienta CASE que permita exportar modelos a codigos fuentes.

Objetivos Específicos

- Seguir el desarrollo sistemático de todas las actividades según el Proceso Unificado , para el software de componente de seguridad.

- Aplicar UML "Lenguaje de Modelado Unificado", para representar todos los diagramas y estereotipos que utiliza el PU en cada flujo de trabajo.

- Refinar los requisitos a partir del conjunto de requerimientos para el software, para esto aplicar el flujo de trabajo análisis según PUDS.
- Proveer una política de administración simple y eficiente.

REQUERIMIENTOS

Modelo de dominio

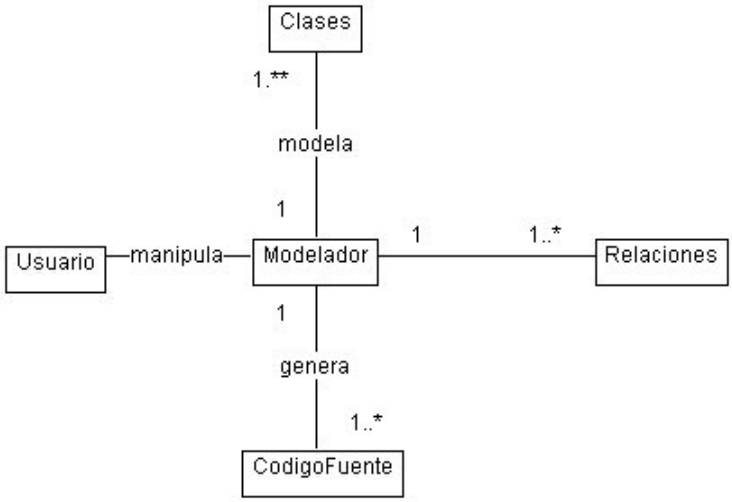

Ilustración 1: modelo de dominio

Modelo de negocio

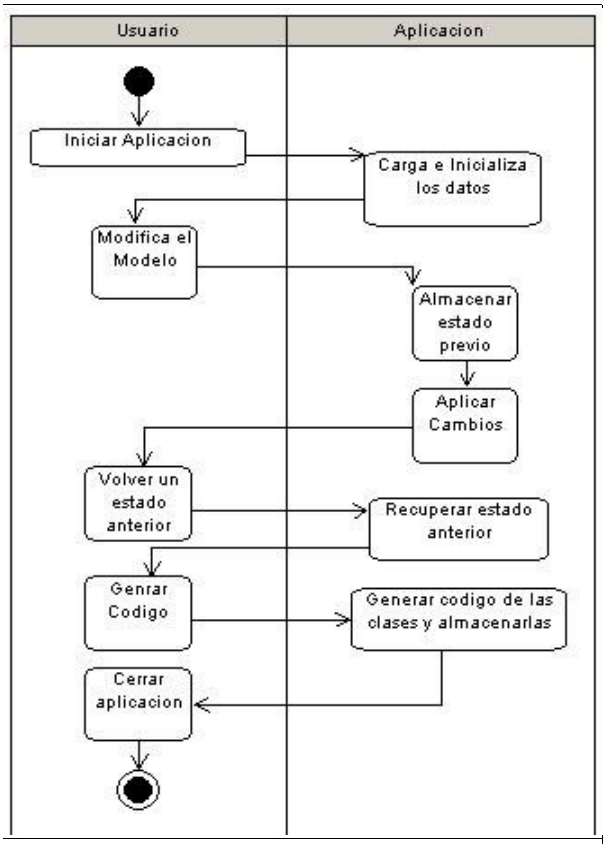

Ilustración 2: Modelo negocio

REQUISITOS

Identificación de actores y casos de uso

Actores

Usuario: Cliente que modifica el modelo de clases, genera las relaciones, modifica las clases y genera código.

Casos de Uso

- Gestionar clases : Permite agregar, eliminar y modificar las propiedades de las clases del modelo.
- Gestionar relaciones : Permite agregar, eliminar y modificar las propiedades de las

relaciones del modelo.

- Gestionar estados : Permite retornar el modelo a un estado previo.
- Generar código : Permite generar código del modelo, segun algun lenguaje elegido.

Priorización de casos de uso

Caso de Uso	Prioridad	Estado
Gestionar clases	Alta	Aprobado
Gestionar relaciones	Alta	Aprobado
Gestionar estados	Media	Aprobado
Generar código	Media	Aprobado

Tabla 1: Priorizacion casos de uso

Detallar casos de uso

Caso de uso : Gestionar clases

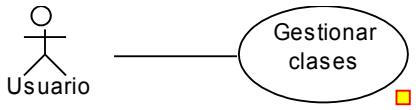

Ilustración 3: CU gestionar clases

Caso de Uso	Gestionar clases
Identificador	0
Actores	Usuario
Propósito	Permite agregar, eliminar y modificar las propiedades de las clases del modelo.
Iniciador	Usuario
Precondiciones	Ninguna
Flujo Principal	1. El usuario inicia la aplicación. 2. El usuario agrega una clase al modelo. 3. La aplicacion verifica si el nombre de la clase es duplicado. a) Si no existe, lo agrega al modelo. b) Si existe, no lo agrega. 4. El usuario modifica las propiedades de una clase 5. Se inicia el dialogo de propiedades de la clase seleccionada. 6. Una vez aceptadas las modificaciones, se actualiza la clase con los nuevos valores. 7. El usuario elimina una clase del modelo 8. La aplicacion elimina todas las relaciones que lo contengan y actualiza el modelo.
Postcondiciones	Ninguna
Flujos alternativos	1. Ninguno.

Tabla 2: Detalle CU-0

Caso de uso : Gestionar relaciones

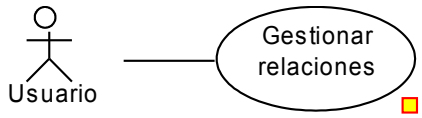

Ilustración 4: Gestoinar relaciones

Caso de Uso	Gestionar Relaciones
Identificador	1
Actores	Usuario
Propósito	Permite agregar, eliminar y modificar las propiedades de las relaciones del modelo.
Iniciador	Usuario
Precondiciones	Ninguna
Flujo Principal	9. El usuario inicia la aplicación. 10. El usuario agrega una clase al modelo. 11. La aplicacion verifica si el nombre de la relacion es duplicado. a) Si no existe, lo agrega al modelo. b) Si existe, no lo agrega. 12. El usuario modifica las propiedades de una relacion. 13. Se inicia el dialogo de propiedades de la clase seleccionada. 14. Una vez aceptadas las modificaciones, se actualiza la clase con los nuevos valores. 15. El usuario elimina una clase del modelo 16. La aplicacion elimina todas las relaciones que lo contengan y actualiza el modelo.
Postcondiciones	Ninguna
Flujos alternativos	1. Ninguno.

Tabla 3: CU 1

Caso de uso : Gestionar Estados

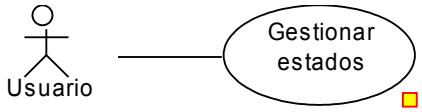

Ilustración 5: CU1 gestionar estados

Caso de Uso	Gestionar Estados
Identificador	2
Actores	Gestor proyecto, desarrollador
Propósito	Permitir tener los estados en contacto con la aplicacion los estados deben ser para mantener avanzar o retroceder un ¡a accion.
Iniciador	Usuario
Precondiciones	Gestionar Usuarios,Gestionar Relaciones
Flujo Principal	1. El usuario inicia la aplicación cliente. 2. El sistema se pide indicar el anterior estado de la aplicaion o el siguiente estado. 3. Mantiene un valor del estado anterior para luego ser devuelto a la aplicacion. 4. Si el usuario inicia un estado. a) Se le otorgan un solo accionde volver atras. 5. Sino a) Se le otroga ir hacia adelante una vez retrocedido a un estado anterior 6. La aplicación obtiene los datos referentes al modelo y los visualiza. 7. El usuario finaliza la aplicación cuando cierra todos los estado. 8. La aplicación envia un mensaje de finalización y espera la aprobación del gestor de proyecto para finalizar su ejecución.
Postcondiciones	Ninguna
Flujos alternativos	1. Si no se ha establecido un estado anterior la aplicacion no respondera por mas que haga varios intentos de volver atras por que esta queda los estado en nulo.

Tabla 4: Detalle CU-2

Caso de Uso : Generar Codigo

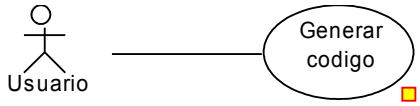

Ilustración 6: CU2 generar codigo

Caso de Uso	Generar codigo
Identificador	3
Actores	Usuario
Propósito	Permite generar codigo fuente para un lenguaje de programacion en este caso java.
Iniciador	Usuario
Precondiciones	Ninguna
Flujo Principal	1. El usuario utiliza la aplicacion para lo cual diseña un diagrama para las especificacion ya antes mencionadas como agregacion, compocicion, generalizacion , asociacio. 2. La aplicacion reconoce el tipo de relacion y la cantidad de clases. 3. La aplicacion permite la configuracion de si el usuario quiere o no implementar los selectores y ponedores de las clase. 4. Sino a) Solo se carganara los metodo si es que los puso dentro de laclase. 5. La aplicacion reconocera cada una de las clase para saber si es una clase, interfaz, o una clase abstracta. 6. Teniendo esa informacion la plaicacion utiliza un Buffer de escriura para ese tipo de clase y lo genera un una carpeta prederminada por el programador. 7. Generara para cada una de las clases con sus respectivas relaciones y con los atributos correspondienres.
Postcondiciones	Ninguna
Flujos alt.	

Tabla 5: Detalle CU-3

ANÁLISIS

Análisis de arquitectura

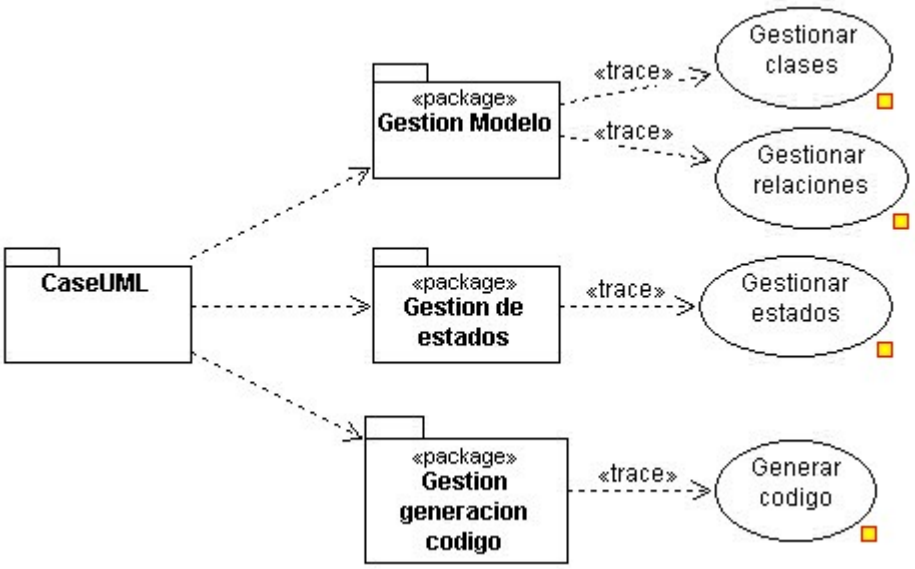

Ilustración 7: Analisis de arquitectura

Análisis de Casos de Uso

Caso de Uso : Gestionar clases

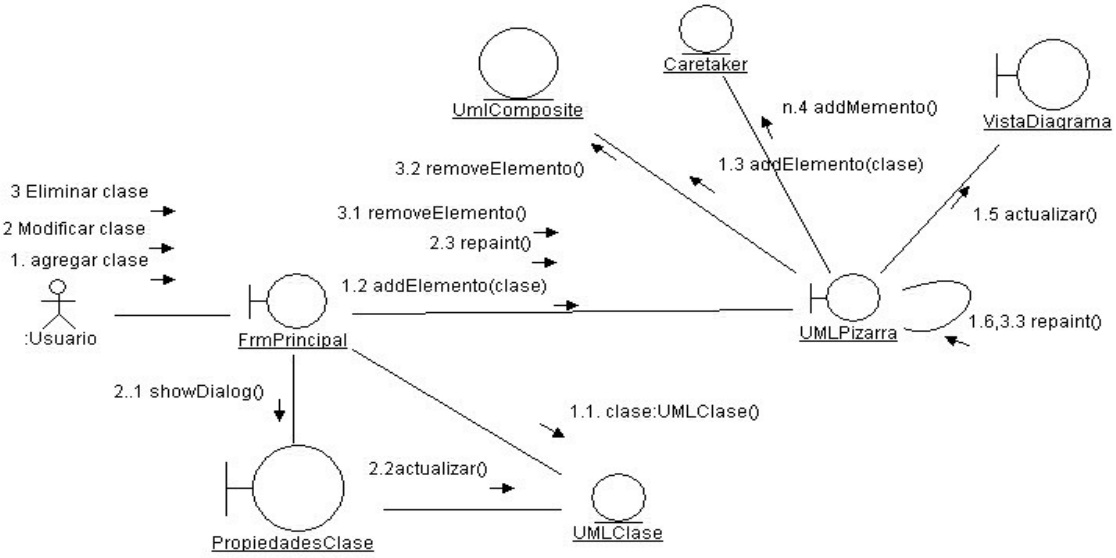

Ilustración 8: Diag. colaboracion, gestionar clases

Caso de Uso : Gestoinar relaciones

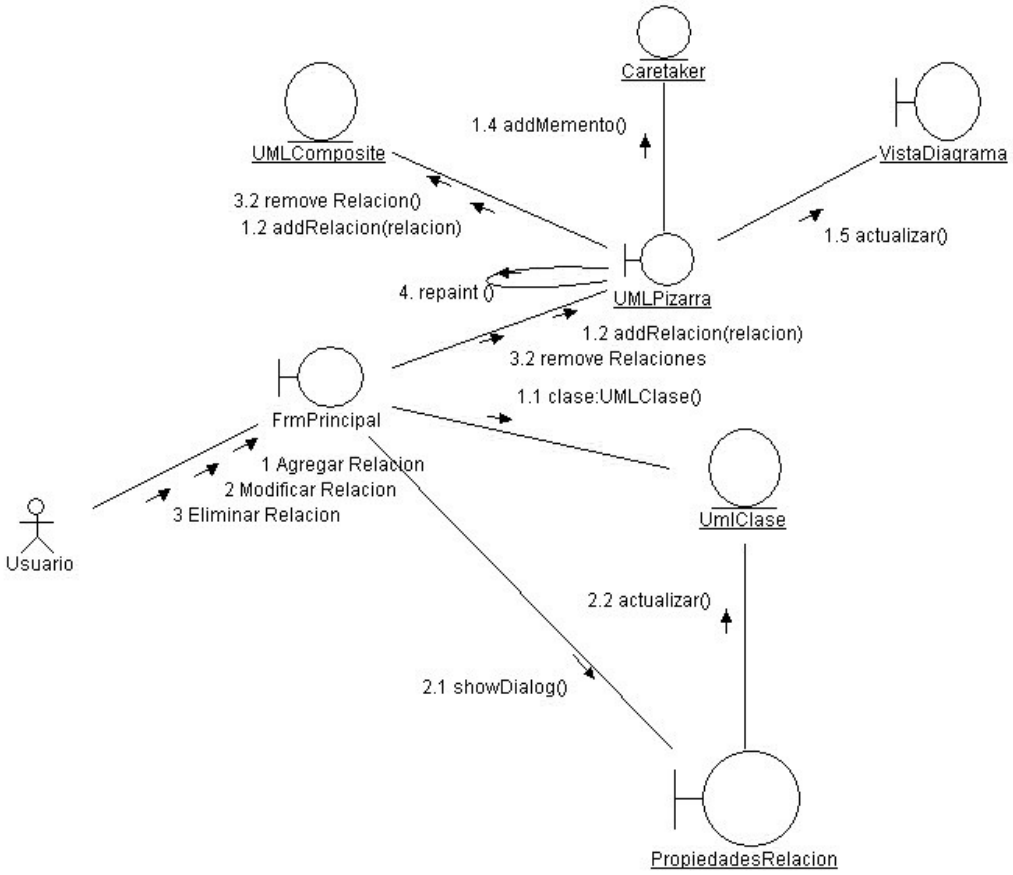

Ilustración 9: Diag.colaboracion, gestionar relaciones

Caso de Uso : Gestionar estados

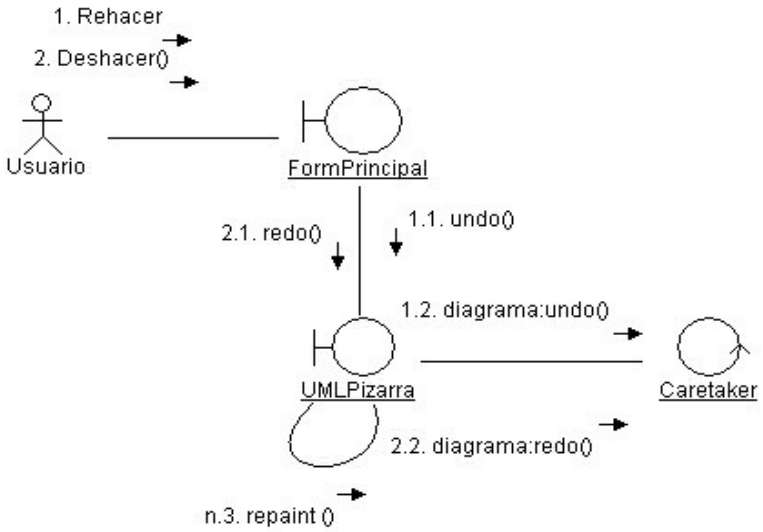

Ilustración 10: Diag.colaboracion, gestionar estados

Caso de Uso : Generar codigo fuente

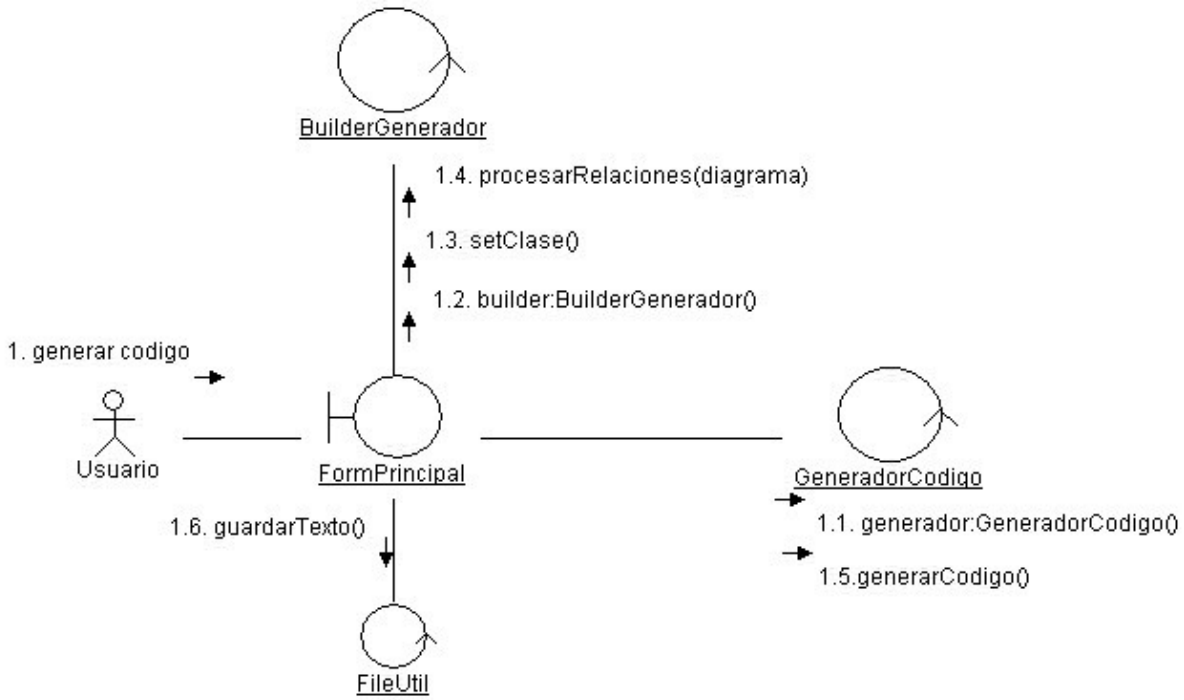

Ilustración 11: Diag.colaboracion,generar codigo fuente

Análisis de Clases

Clases Interfaz

FormPrincipal

Nombre	FormPrincipal	
Tipo	Formulario/Diálogo	
Propósito	Visualizar el modelo y los elementos.	
Atributos	Identificador	Tipo
	PanelDiagrama	panel
	barraTareas	Barra
Operaciones	Cerrar	
Observaciones		

PropiedadesClase

Nombre	PropiedadesClase	
Tipo	Formulario/Diálogo	
Propósito	Permitir modificar las propiedades de una clase	
Atributos	Identificador	Tipo
	General	panel
	Atributos	panel
	Metodos	panel
Operaciones	Aceptar,Cancelar	
Observaciones		

PropiedadesRelacion

Nombre	PropiedadesRelacion	
Tipo	Formulario/Diálogo	
Propósito	Permitir la edición visual delas relaciones.	
Atributos	Identificador	Tipo
	barraTareas	Barra de tareas

	general	panel
Operaciones	Aceptar,Cancelar	
Observaciones		

Clases de Control

UMLPizarra

Nombre	UMLPizarra
Propósito	Administra los elementos del diagrama, el manejador de estados y las vistas, es la clase controlador del MVC.
Entradas	

Caretaker

Nombre	Caretaker
Propósito	Encargado de almacenar y manipular los estados del diagrama, permite regresar a un estado anterior(Memento).
Entradas	Diagrama, modelo

BuilderGenerador

Nombre	BuilderGenerador
Propósito	Clase que inicializa y construye los generadores de codigo, facilitando su construcción.
Entradas	

GeneradorCodigo

Nombre	GeneradorCodigo
Propósito	Interfaz que define los metodos necesarios para generar codigo fuente de una clase.
Entradas	UMLClase

Clases Entidad

UMLElemento

Nombre	UMLElemento	
Responsabilidad	Clase base UML, todos los elementos denominados UML deben especializarla.	
Atributos	Identificador	Tipo
	Nombre	cadena
	Visibilidad	byte
	Estereotipo	cadena
Relación	Ninguna	

UMLClase

Nombre	UMLClase	
Responsabilidad	Especializacion de UMLElemento, especifica una clase de UML.	
Atributos	Identificador	Tipo
	Metodos	Metodo
Relación	Metodo	

UMLRelacion

Nombre	UMLRelacion	
Responsabilidad	Clase que relaciona dos elementos UML cualquiera.	
Atributos	Identificador	Tipo
	Origen	UMLElemento
	destino	UMLElemento
Relación		

UMLComposite

Nombre	UMLComposite
Responsabilidad	Clase compuesta que hereda de UMLRelacion, contiene a todos los elementos del diagrama.

Atributos	Elementos
Relación	Ninguna

Metodo

Nombre	Metodo
Responsabilidad	Clase que representa un metodo o funcion.
Atributos	Parametros,nombre,salida,acceso
Relación	Ninguna

Análisis de paquetes

Gestión de modelo

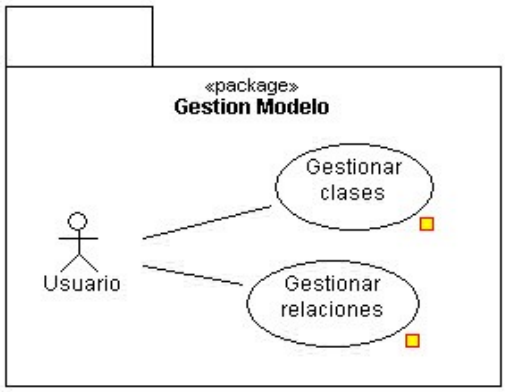

Ilustración 12: Gestion de modelo

Gestión de estados

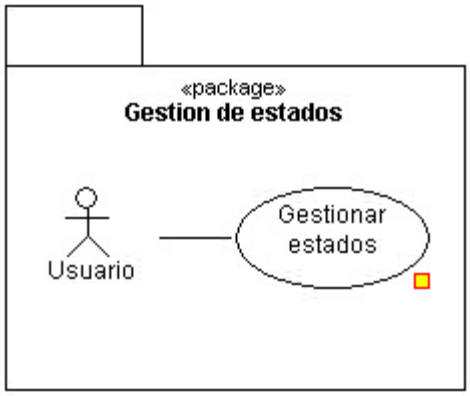

Ilustración 13: Gestion de estados

Generar codigo

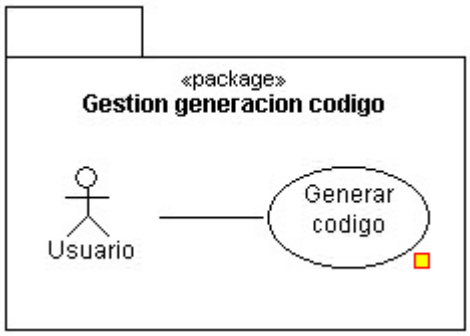

Ilustración 14: Generar codigo

DISEÑO

Diseño de Arquitectura

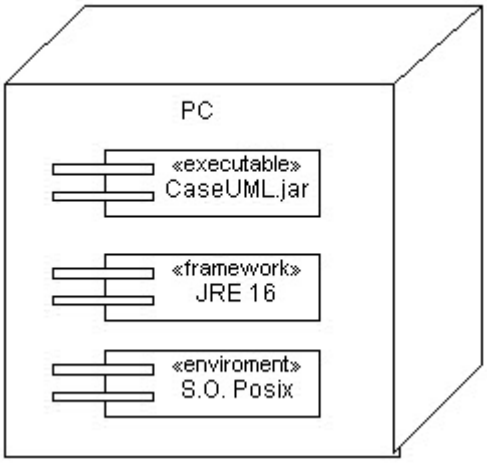

Ilustración 15: Diagrama de despliegue

Diseño de Casos de Uso

Caso de Uso: Gestionar clases

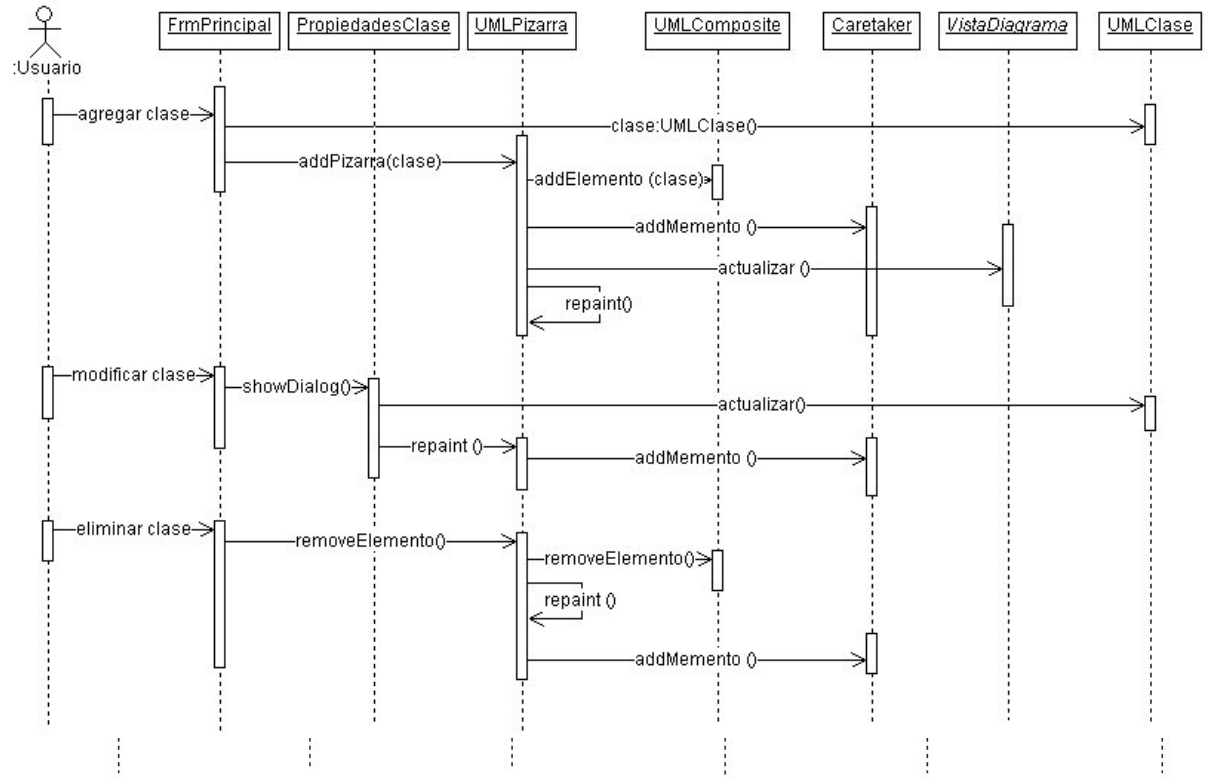

Ilustración 16: Diag.secuencia,gestionar sesion

Caso de Uso : Gestionar relaciones

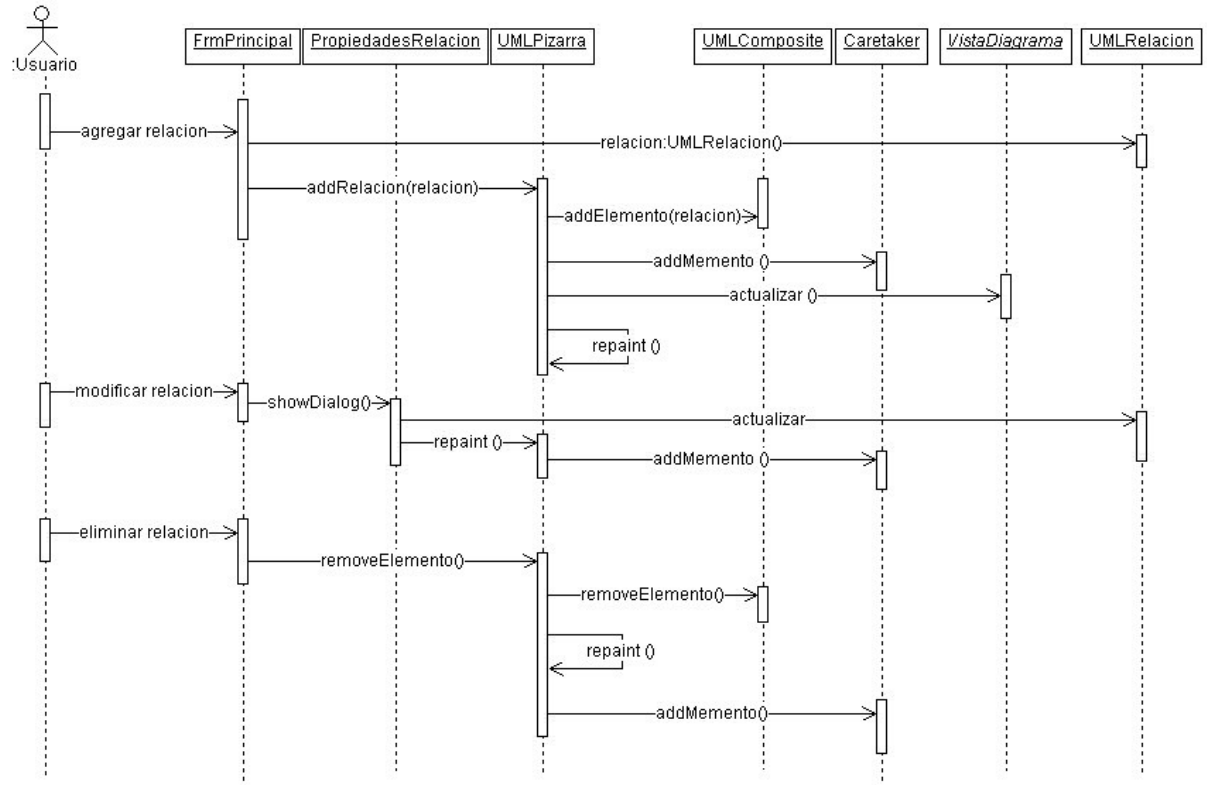

Ilustración 18: Diag.secuencia,gestionar relaciones

Caso de Uso : Gestionar estados

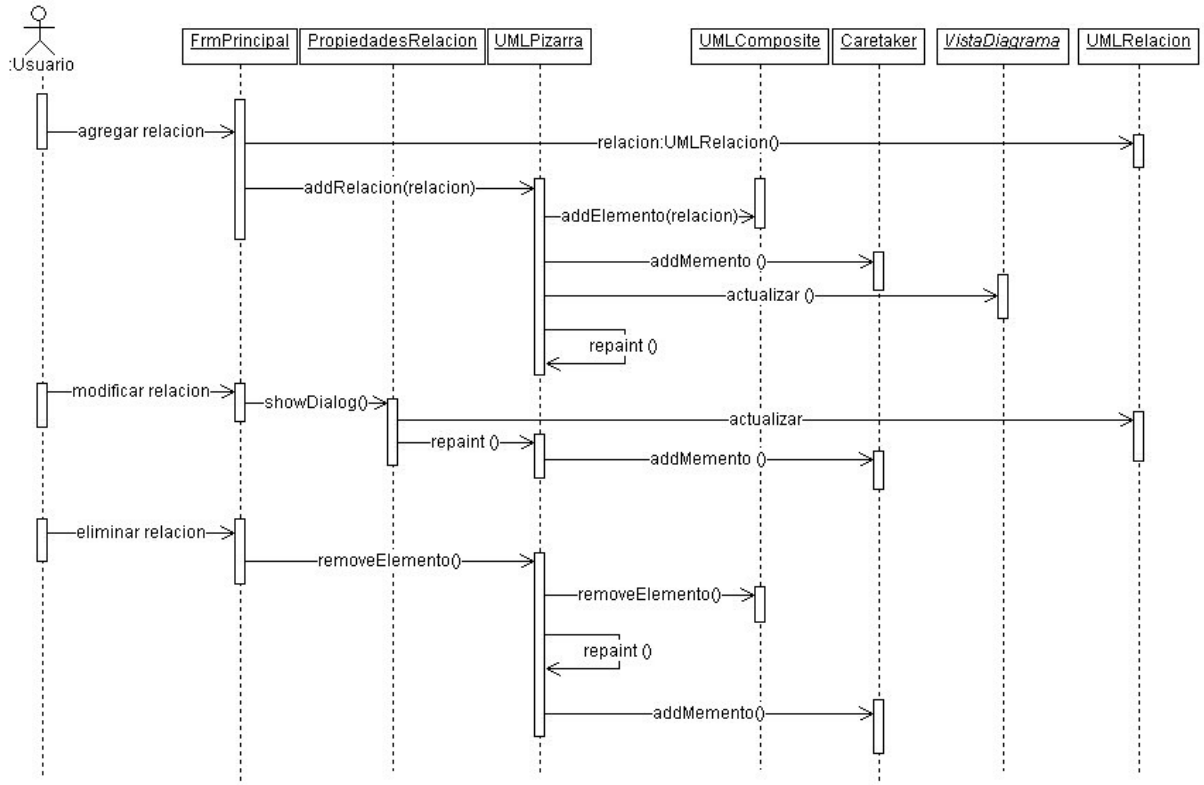

Ilustración 19: Diag.secuencia,gestionar estados

Caso de Uso : Generar codigo

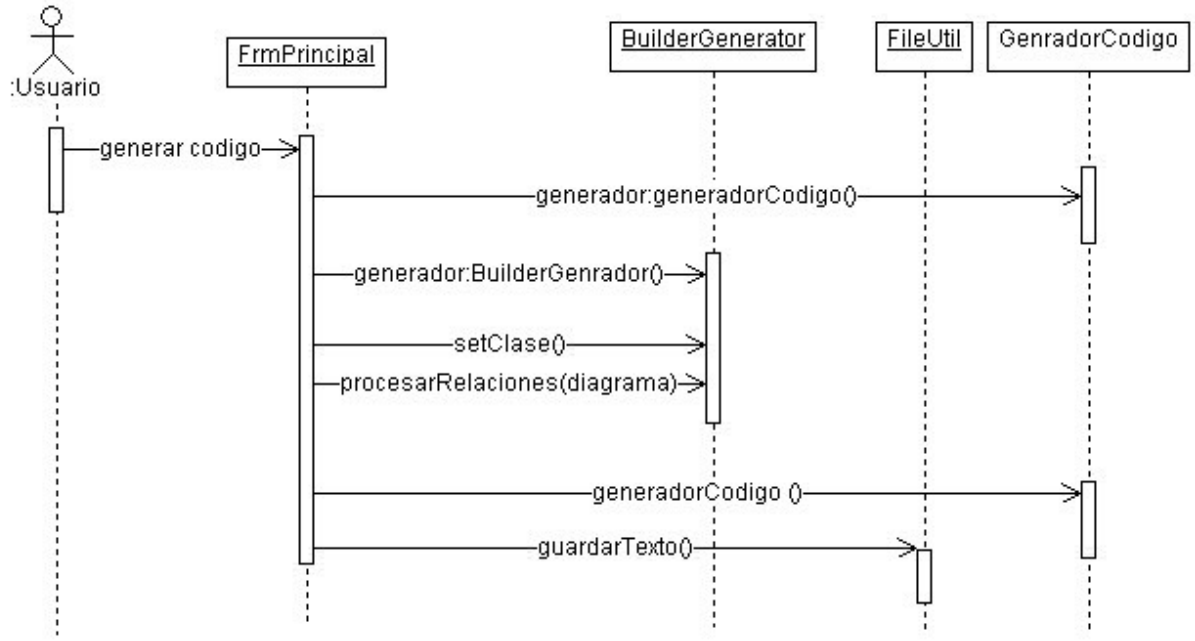

Ilustración 20: Diag.secuencia,generar codigo

Diseño de Clases

Diseño de clases

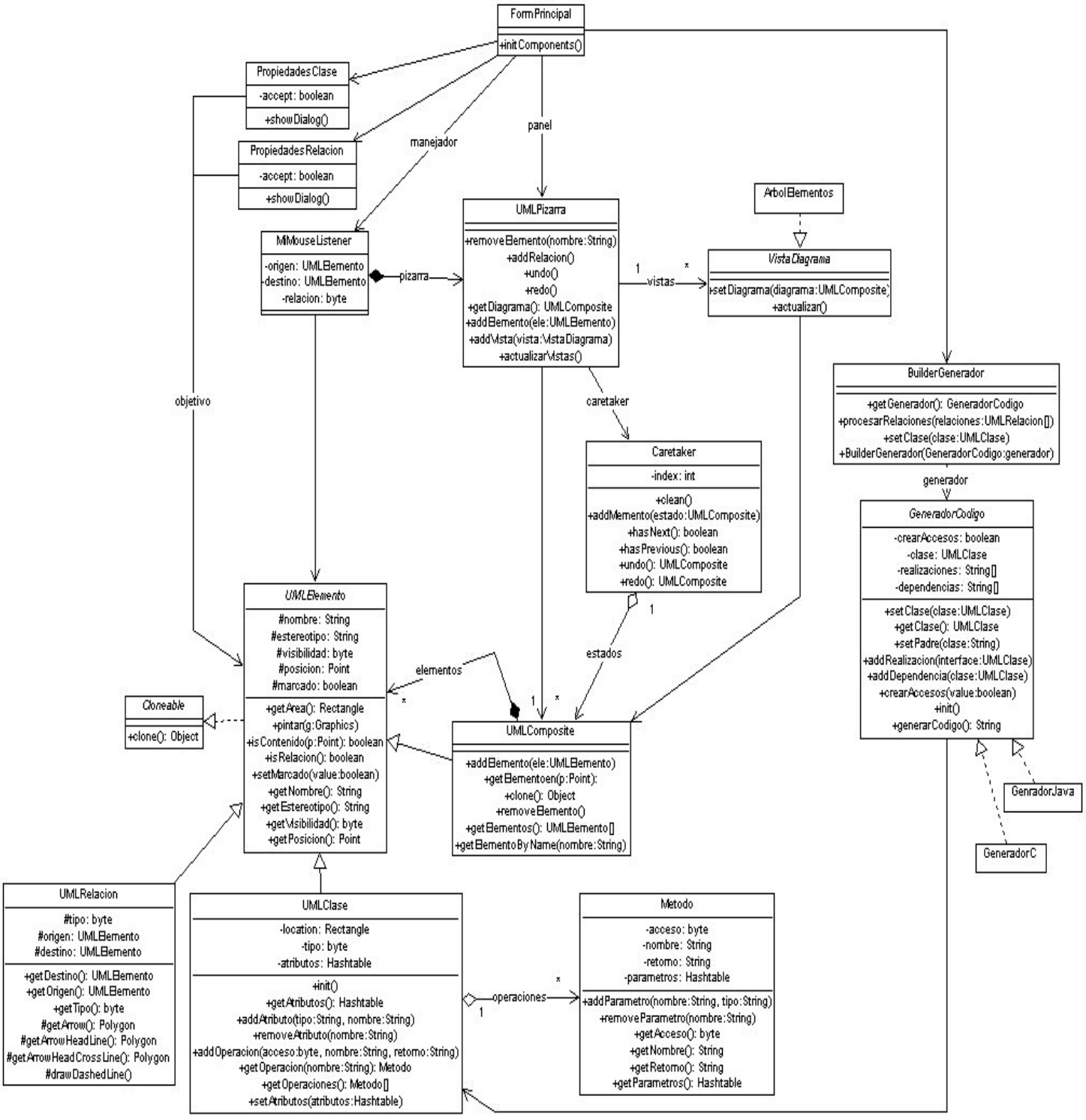

Ilustración 22: Diag.clases,aplicacion

IMPLEMENTACIÓN

Implementación de la Arquitectura

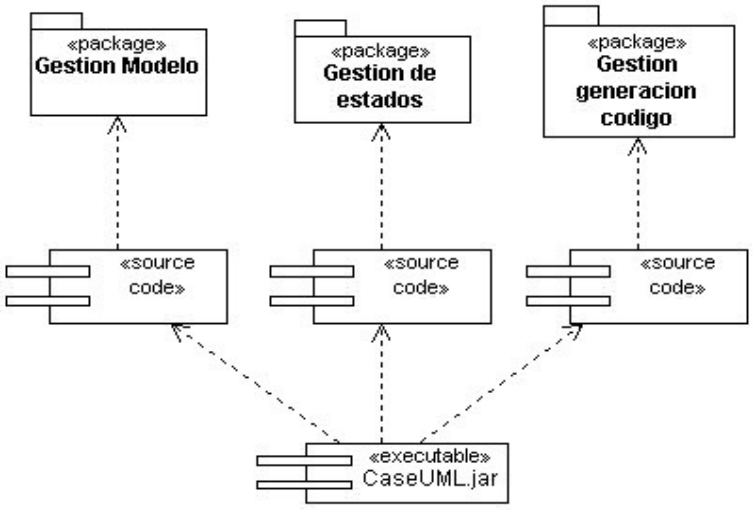

Ilustración 23: Implementacion de la arquitectura

Implementación de Subsistemas

Gestión modelo

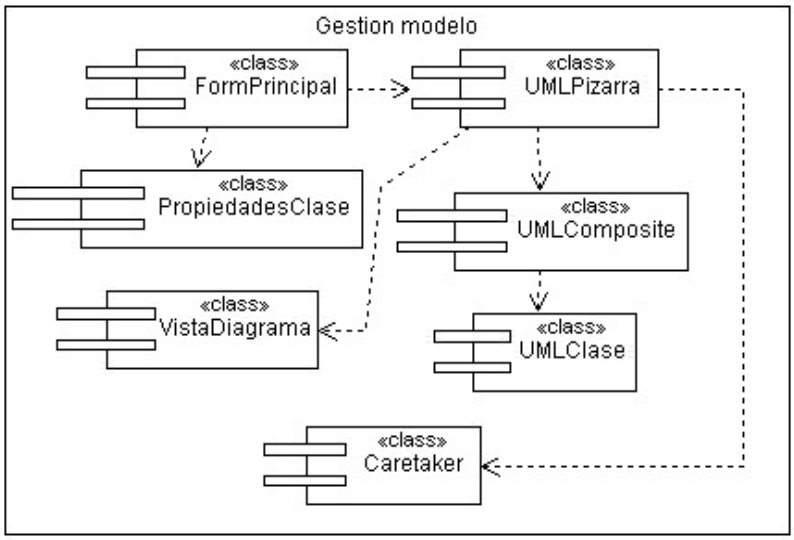

Ilustración 24: Diag.comp, gestion modelo

Gestión de estados

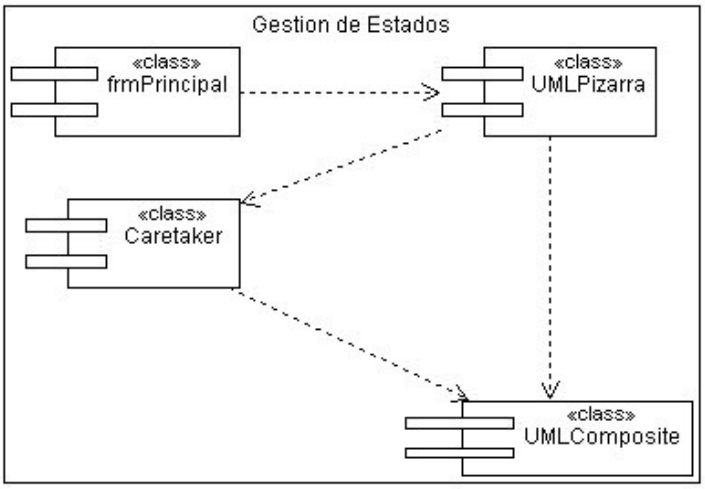

Ilustración 25: Diag.comp,gestion estados

Gestión generación de codigo

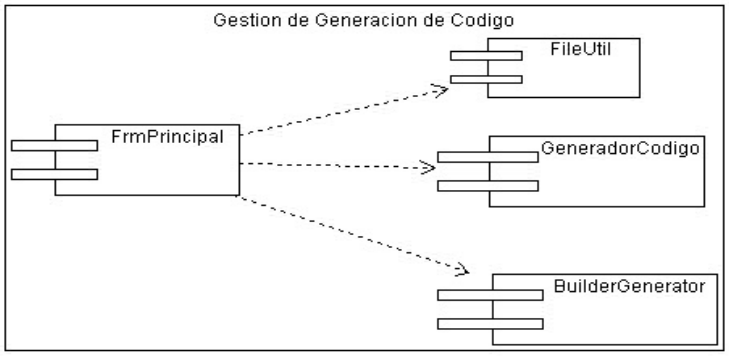

Ilustración 26: Diag.comp,gestion generación de codigo