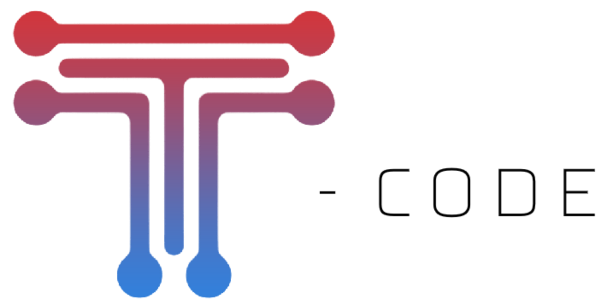
Modelo de Diseño

Vesta Risk Manager

T-Code

Agustín Collareda, Cintia Hernandez, Hugo Frey



El propósito del Modelo de Diseño es empezar a realizar los casos de uso desarrollados durante las etapas anteriores en especial la etapa de captura de requerimientos. Es decir, tomar el Modelo de Casos de Uso y las Especificaciones Suplementarias creadas con anterioridad entre otros insumos y generar un modelo de diseño que pueda ser usado por los desarrolladores durante la etapa de implementación.



Tabla de contenido

[Introducción 5](#_Toc181722956)

[Propósito 5](#_Toc181722957)

[Alcance 5](#_Toc181722958)

[Referencias 5](#_Toc181722959)

[Visión general 6](#_Toc181722960)

[Diseño de Casos de Uso 6](#_Toc181722961)

[Caso de Uso 1: Autentificarse 6](#_Toc181722962)

[Especificación de caso de uso de diseño 6](#_Toc181722963)

[Diagrama de paquetes 6](#_Toc181722964)

[Diagrama de Interacción 7](#_Toc181722965)

[Requerimientos especiales o de implementación 7](#_Toc181722966)

[Caso de Uso 2: Administrar acceso al sistema 7](#_Toc181722967)

[Especificación de caso de uso de diseño 7](#_Toc181722968)

[Diagrama de paquetes 9](#_Toc181722969)

[Diagrama de Interacción 11](#_Toc181722970)

[Caso de Uso 3: Administrar proyectos 12](#_Toc181722971)

[Especificación de caso de uso de diseño 12](#_Toc181722972)

[Diagrama de paquetes 13](#_Toc181722973)

[Diagrama de Interacción 15](#_Toc181722974)

[Caso de Uso 4: Añadir riesgo a la lista 16](#_Toc181722975)

[Especificación de caso de uso de diseño 16](#_Toc181722976)

[Diagrama de paquetes 16](#_Toc181722977)

[Diagrama de Interacción 18](#_Toc181722978)

[Caso de Uso 5: Modificar lista de riesgos 18](#_Toc181722979)

[Especificación de caso de uso de diseño 18](#_Toc181722980)

[Diagrama de paquetes 19](#_Toc181722981)

[Diagrama de Interacción 21](#_Toc181722982)

[Caso de Uso 6: Administrar categorías de riesgos 22](#_Toc181722983)

[Especificación de caso de uso de diseño 22](#_Toc181722984)

[Diagrama de paquetes 22](#_Toc181722985)

[Diagrama de Interacción 23](#_Toc181722986)

[Caso de Uso 7: Realizar evaluación de riesgo 24](#_Toc181722987)

[Especificación de caso de uso de diseño 24](#_Toc181722988)

[Diagrama de paquetes 24](#_Toc181722989)

[Diagrama de Interacción 26](#_Toc181722990)

[Caso de Uso 8: Añadir plan de riesgo 26](#_Toc181722991)

[Especificación de caso de uso de diseño 26](#_Toc181722992)

[Diagrama de paquetes 27](#_Toc181722993)

[Diagrama de Interacción 29](#_Toc181722994)

[Caso de Uso 9: Modificar plan de acción 29](#_Toc181722995)

[Especificación de caso de uso de diseño 29](#_Toc181722996)

[Diagrama de paquetes 30](#_Toc181722997)

[Diagrama de Interacción 32](#_Toc181722998)

[Caso de Uso 10: Realizar informes 32](#_Toc181722999)

[Especificación de caso de uso de diseño 32](#_Toc181723000)

[Diagrama de paquetes 33](#_Toc181723001)

[Diagrama de Interacción 35](#_Toc181723002)

[Caso de Uso 11: Exportar archivos 36](#_Toc181723003)

[Especificación de caso de uso de diseño 36](#_Toc181723004)

[Diagrama de paquetes 36](#_Toc181723005)

[Diagrama de Interacción 37](#_Toc181723006)

[Caso de Uso 12: Realizar análisis de riesgo 37](#_Toc181723007)

[Especificación de caso de uso de diseño 37](#_Toc181723008)

[Diagrama de paquetes 38](#_Toc181723009)

[Diagrama de Interacción 40](#_Toc181723010)

[Diagramas 41](#_Toc181723011)

[Diagrama de componentes 41](#_Toc181723012)

[Diagrama de Clases 42](#_Toc181723013)

[Diagramas de Paquetes 43](#_Toc181723014)

Modelo de Diseño

Introducción

El presente documento detalla el modelo de diseño para nuestro proyecto, Vesta Risk Manger. Este documento es una pieza clave en el proceso de desarrollo, ya que proporciona una visión integral de la arquitectura, componentes y estructura general del sistema que estamos construyendo. Además, este documento actúa como una guía invaluable durante la fase de implementación. Proporciona a los desarrolladores una hoja de ruta clara, reduciendo la posibilidad de malentendidos y errores en la construcción del sistema.

Propósito

El propósito principal de este documento es proporcionar una visión clara y completa del diseño de Vesta Risk Manager. Sirve como un punto de referencia para todos los miembros del equipo de desarrollo, diseñadores, y stakeholders involucrados en el proyecto.

Alcance

Este documento de Modelo de Diseño abarca la totalidad de Vesta Risk Manager, incluyendo todos sus componentes, módulos y funcionalidades. El alcance se extiende a la arquitectura general del sistema, el diseño de la interfaz de usuario, la estructura de la base de datos, los componentes del back-end, las integraciones con sistemas externos, los flujos de datos y procesos principales, así como las consideraciones de seguridad y rendimiento.

Referencias

Los documentos a los que se le hace referencia en este documento son:

* Propuesta de desarrollo.
* Especificación de requerimientos.
* Modelo de casos de uso.
* Modelo de datos.
* Arquitectura del sistema.

Estos documentos fueron realizados por el equipo de T-Code.

Visión general

Diseño de Casos de Uso

Caso de Uso 1: Autentificarse

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°1** | **Autentificarse** | |
| Actor Principal | Administrador del sistema; Líder del proyecto; Desarrollador | |
| Precondiciones | El actor ingresa a la página index del sistema | |
| Disparador | El actor selecciona “iniciar sesión” en la página de index | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor inicia sesión con su cuenta de Google. |
| 2 | La InterfazUsuariosolicita *iniciarSesion(googleUser)* al GestorUsuario. |
| 3 | El GestorUsuario envía el evento *verificarUsuario(email)* a Usuario, Usuario solicita *verificarUsuario(email)* a la BDD, la BDD verifica el usuario, la BDD retorna la respuesta a Usuario, Usuario retorna la respuesta al GestorUsuario (3a). El GestorUsuario retorna la respuesta a la InterfazUsuario. La InterfazUsuario redirige al actor a la página de inicio del sistema. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 3a | El actor no se encuentra registrado en la base de datos. El GestorUsuario retorna la respuesta a la InterfazUsuario. La interfazUsuario redirige al actor a la página index del sistema y muestra un mensaje de error. |

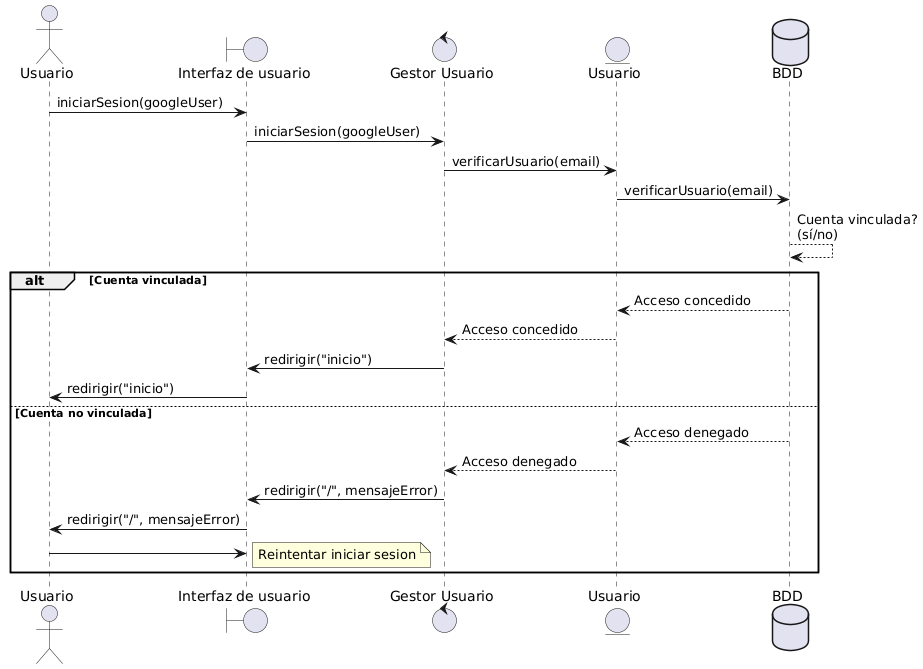
Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

* GestorUsuario.
  + Tendrá los métodos: Constructor, retornar un usuario por correo.
  + Comentario: El método obtener usuario tendrá que contar con un método privado que valide que se está pasando un correo.
* Usuario.
  + Tendrá los atributos de nombre, correo y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, retornar un usuario por correo.

Estas clases están vinculada al subsistema de UARGflow.

Diagrama de Interacción



**Fig. 1.1**: Diagrama de secuencia CU01.

Requerimientos especiales o de implementación

* **(RNF04):** El sistema deberá ser integrado con UARGflow para el inicio de sesión de los usuarios.

Caso de Uso 2: Administrar acceso al sistema

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°2** | **Administrar acceso al sistema** | |
| Actor Principal | Administrador | |
| Precondiciones | El usuario inicio sesión en el sistema | |
| Disparador | El usuario selecciona una de las opciones de administración de acceso | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor ingresa a la lista de usuarios. |
|  | 2 | La InterfazUsuario solicita *listarUsuarios* al GestorUsuario. El GestorUsuario envía el evento *listarUsuarios* a Usuario. Usuario solicita *listarUsuarios* a la BDD, la BDD realiza la transacción, la BDD retorna la respuesta al Usuario, el Usuario retorna la respuesta al GestorUsuario. |
|  | 3 | El GestorUsuario retorna los usuarios a la InterfazUsuario. La InterfazUsuario redirige al actor a la página /usuarios. |
|  | 4 | El actor selecciona “Registrar usuario” (4a). El actor selecciona “Eliminar usuario” (4b). El actor selecciona “Editar Usuario” (4c). |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 4a | 1. La InterfazUsuario redirige al actor a la página /usuarios/crear. 2. El actor ingresa los datos del usuario a *registrar (nombre, email, roles)* en el formulario de la página y selecciona confirmar. Si selecciona cancelar, la InterfazUsuario redirige al actor a /usuarios. 3. La InterfazUsuario solicita *crearUsuario(datos)* a GestorUsuario. El GestorUsuario comprueba los datos y envía el evento *crearUsuario(datos)* a Usuario. Usuario solicita *crearUsuario(datos)* a la BDD. La BDD carga el usuario y retorna la respuesta al Usuario, el Usuario retorna la respuesta al GestorUsuario, el GestorUsuario retorna la respuesta a la InterfazUsuario. La InterfazUsuario redirige al actor a la página /usuarios. |
|  | 4b | 1. El actor selecciona el usuario a eliminar. 2. La InterfazUsuario redirige al actor a la página /usuarios/eliminar/:id con la id del usuario seleccionado y solicita confirmación. 3. El actor confirma la eliminación del usuario. Si cancela la eliminación, la InterfazUsuario redirige al actor a /usuarios. 4. La InterfazUsuario envia el evento *eliminarUsuario(id\_usuario)* al GestorUsuario, el GestorUsuario envía el evento *eliminarUsuario(id\_usuario)* a Usuario. Usuario solicita *eliminarUsuario(id\_usuario)* a la BDD. La BDD elimina el usuario. La BDD retorna la respuesta al Usuario, el Usuario retorna la respuesta al GestorUsuario, el GestorUsuario retorna la respuesta a la InterfazUsuario. 5. La InterfazUsuario redirige al actor a /usuarios. |
|  | 4c | 1. El actor selecciona el usuario a modificar. 2. La InterfazUsuario redirige al actor a la página /usuarios/editar/:id con la id del usuario seleccionado. 3. El actor modifica los datos del usuario seleccionado (nombre, email, roles) en el formulario de la página y selecciona confirmar. Si selecciona cancelar, la InterfazUsuario redirige al actor a /usuarios. 4. La InterfazUsuario solicita *editarUsuario(id\_usuario, datos)* a GestorUsuario. El GestorUsuario comprueba los datos y envia el evento *editarUsuario(id\_usuario, datos)* a Usuario. Usuario solicita *editarUsuario(id\_usuario, datos)* a la BDD. La BDD modifica el usuario. La BDD retorna la respuesta al Usuario, el Usuario retorna la respuesta al GestorUsuario, el GestorUsuario retorna a la InterfazUsuario, La InterfazUsuario redirige al actor a /usuarios. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

* GestorUsuario.
  + Tendrá los métodos: Constructor, retornar un usuario por correo, retornar todos los usuarios, modificar un usuario, eliminar un usuario, actualizar perfil de usuario, actualizar permiso de perfil, añadir perfil, eliminar perfil.
  + Comentario: El método obtener usuario tendrá que contar con un método privado que valide que se está pasando un correo, Los métodos que se repiten entre usuario y gestión de usuario es para formatear y respetar con la modularización del sistema.
* Usuario.
  + Tendrá los atributos de nombre, correo, conexión y perfil.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, retornar un usuario por correo, retornar todos los usuarios, modificar un usuario, eliminar un usuario, actualizar perfil de usuario.
* Perfil.
  + Tendrá el atributo nombre, conexión y permisos.
  + Tendrá los métodos: Constructor, método get y set de cada atributo excepto para la conexión, obtener todos los perfiles, obtener un perfil por id, eliminar por id el perfil, actualizar por id el perfil, añadir perfil, actualizar permisos en perfil.
* Permiso.
  + Tendrá el atributo nombre y conexión.
  + Tendrá el método: Constructor, obtener todos los permisos, obtener permisos de usuario.
  + Comentario: Si bien esta clase no aparece en el diagrama de secuencia, esta de manera implícita cuando se habla de administrador, usuario, etc.

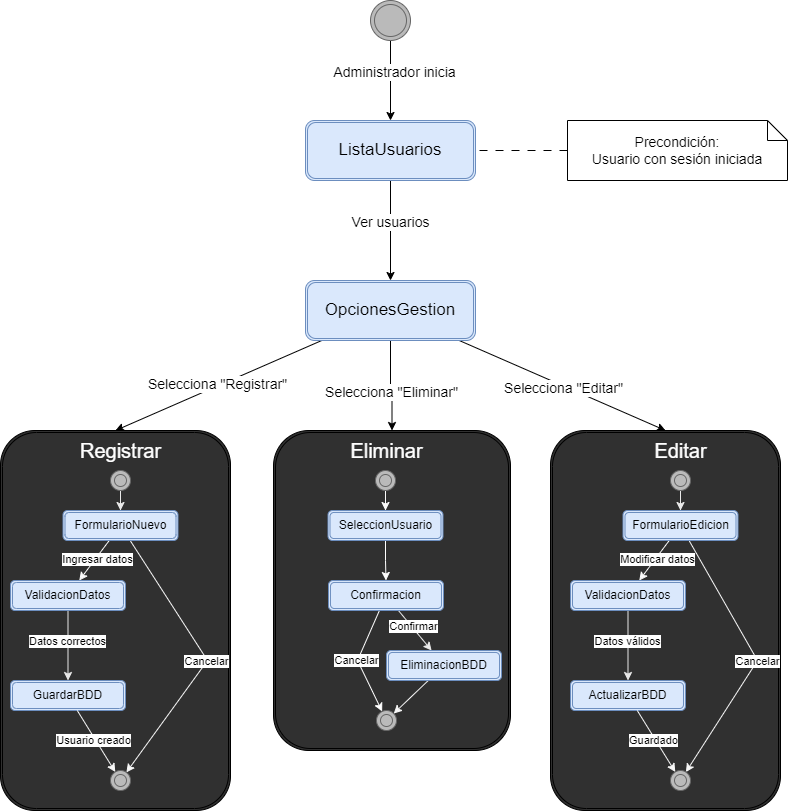
Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción

Diagrama

Descripción generada automáticamente

**Fig. 1.2**: Diagrama de secuencia CU02.



**Fig. 1.3**: Diagrama de flujo CU02.

Caso de Uso 3: Administrar proyectos

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°3** | **Administrar proyectos** | |
| Actor Principal | Administrador | |
| Precondiciones | El usuario inicio sesión en el sistema | |
| Disparador | El usuario selecciona una de las opciones de administración de proyectos | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor ingresa a la lista de proyectos. |
|  | 2 | La InterfazUsuario solicita *listarProyectos* al GestorProyecto. El GestorProyecto envía el evento *listarProyectos* a Proyecto. Proyecto solicita *listarProyectos* a la BDD, la BDD realiza la transacción, la BDD retorna la respuesta al Proyecto, el Proyecto retorna la respuesta al GestorProyecto. El GestorProyecto retorna la respuesta a la InterfazUsuario. La InterfazUsuario redirige al actor a la pagina /proyectos. |
|  | 3 | El actor selecciona "Crear proyecto" (3a). El actor selecciona "Modificar proyecto" (3b). |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 3a | 1. La InterfazUsuario redirige al actor a la página /proyectos/crear. 2. El actor ingresa los datos del proyecto a *registrar (nombre, descripcion, estado, participantes, iteraciones, categorias)* en el formulario de la página y selecciona confirmar. Si selecciona cancelar, la InterfazUsuario redirige al actor a /proyectos. 3. La InterfazUsuario solicita *crearProyecto(datos)* a GestorProyecto. El GestorProyecto comprueba los datos y envía el evento *crearProyecto(datos)* a Proyecto. Proyecto solicita *crearProyecto(datos)* a la BDD. La BDD carga el proyecto y retorna la respuesta al Proyecto, el Proyecto retorna la respuesta al GestorProyecto, el GestorProyecto retorna la respuesta a la InterfazUsuario. La InterfazUsuario redirige al actor a la página /proyectos. |
|  | 3b | 1. El actor selecciona el proyecto a modificar. 2. La InterfazUsuario redirige al actor a la página /proyecto/editar/:id con la id del proyecto seleccionado. 3. El actor modifica los datos del usuario seleccionado (nombre, descripcion, estado, participantes, iteraciones, categorias) en el formulario de la página y selecciona confirmar. Si selecciona cancelar, la InterfazUsuario redirige al actor a /proyectos. 4. La InterfazUsuario solicita *editarProyecto(id\_proyecto, datos)* a GestorProyecto. El GestorProyecto comprueba los datos y envía el evento *editarProyecto(id\_proyecto, datos)* a Proyecto. Proyecto solicita *editarProyecto(id\_proyecto, datos)* a la BDD. La BDD modifica el usuario. La BDD retorna la respuesta al Proyecto, el Proyecto retorna la respuesta al GestorProyecto, el GestorProyecto retorna a la InterfazUsuario, La InterfazUsuario redirige al actor a /usuarios. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

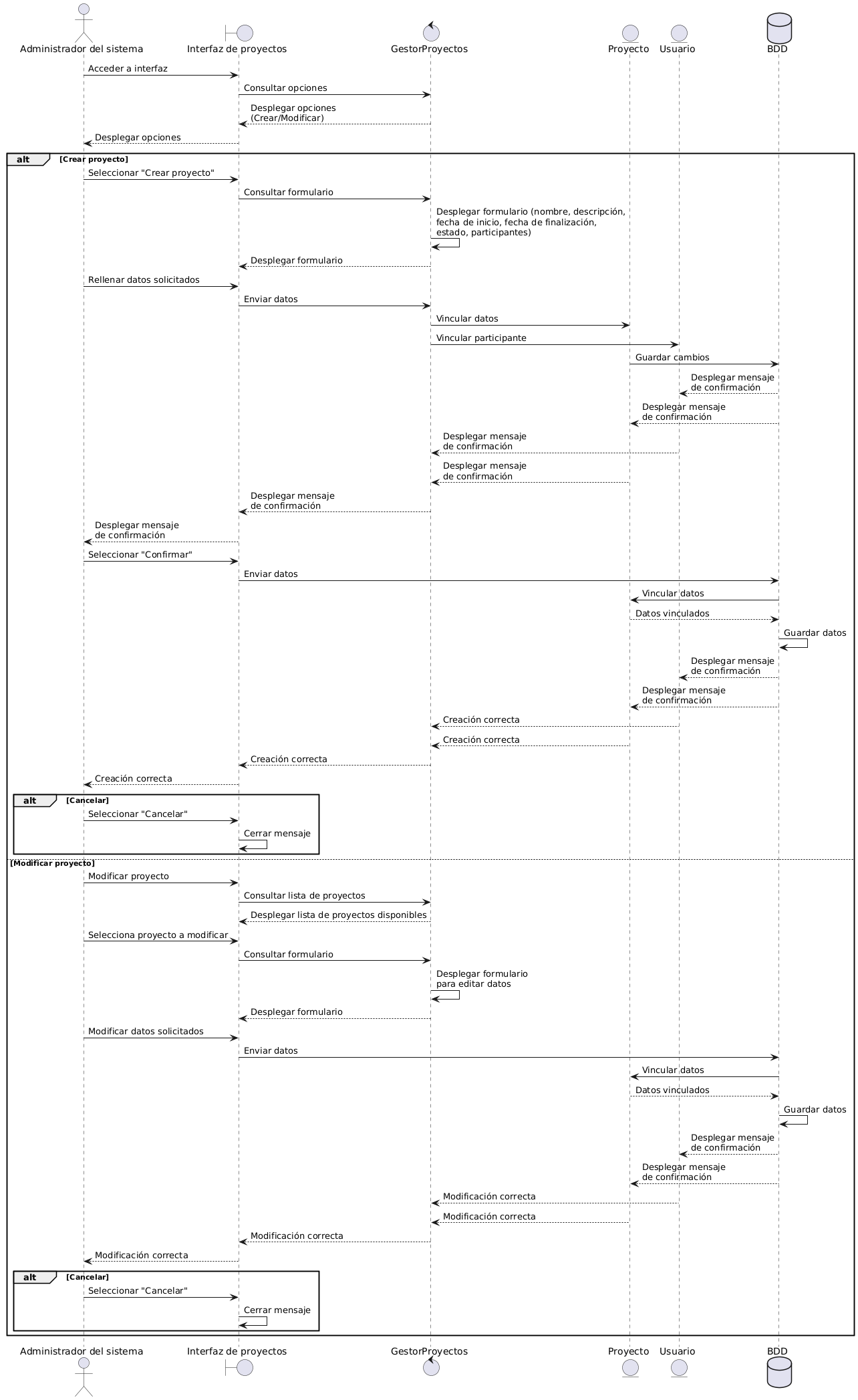
* GestorProyecto.
  + Tendrá los métodos: Constructor, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto, vincular o desvincular iteraciones, obtener todos los proyectos, vincular o desvincular participantes, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.
  + Comentario: Cada método deberá comprobar que no se pase ningún campo vacío, ni ninguna inyección sql.
* Proyecto.
  + Tendrá los atributos de nombre, descripción, estado, fecha inicio, fecha fin, iteraciones, participantes y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto por id, actualizar iteraciones, obtener todos los proyectos, vincular o desvincular participantes.
* Iteracion.
  + Tendrá los atributos fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, método get y set de cada atributo excepto para la conexión, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.

Estas clases están vinculadas al subsistema de Proyectos.

* GestorUsuario.
  + Tendrá el método: Constructor, obtener todos los usuarios.
* Usuario
  + Tendrá los atributos nombre, correo y conexión.
  + Tendrá los métodos: Constructor, obtener todos los usuarios.

Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción



**Fig. 1.4**: Diagrama de secuencia CU03.



**Fig. 1.5**: Diagrama de flujo CU03.

Caso de Uso 4: Añadir riesgo a la lista

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°4** | **Añadir riesgo a la lista** | |
| Actor Principal | Líder de proyecto y Desarrollador | |
| Precondiciones | El usuario inicio sesión en el sistema. El usuario está vinculado a un proyecto | |
| Disparador | El usuario ingresa al proyecto y selecciona añadir riesgo. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona “Añadir riesgo”. |
|  | 2 | La InterfazUsuario redirige al actor a la página /proyecto/:nombre/riesgo/crear/. |
|  | 3 | El actor ingresa los datos del riesgo (descripción, responsable, categoría) en el formulario de la página. El actor selecciona confirmar. Si selecciona cancelar, la InterfazUsuario redirige al actor a la página /proyecto/:nombre/riesgo. |
|  | 4 | La InterfazUsuario solicita *crearRiesgo(datos)* a GestorRiesgo. El GestorRiesgo envia el evento *crearRiesgo(datos)* a Riesgo. Riesgo solicita *crearRiesgo(datos)* a la BDD. La BDD carga el riesgo. La BDD retorna la respuesta a Riesgo. El Riesgo retorna la respuesta al GestorRiesgo, el GestorRiesgo retorna la respuesta a la InterfazUsuario. La InterfazUsuario redirige al actor a /proyecto/:nombre/riesgo. |
| EXTENSIONES | Paso | Acción de Derivación |
|  |  |  |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.

Estas clases están vinculadas al subsistema de Riesgos.

* GestorProyecto.
  + Tendrá los métodos: Constructor, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto, vincular o desvincular iteraciones, obtener todos los proyectos, vincular o desvincular participantes, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.
  + Comentario: Cada método deberá comprobar que no se pase ningún campo vacío, ni ninguna inyección sql.
* Proyecto.
  + Tendrá los atributos de nombre, descripción, estado, fecha inicio, fecha fin, iteraciones, participantes y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto, actualizar iteraciones, obtener todos los proyectos, vincular o desvincular participantes.
* Iteracion.
  + Tendrá los atributos fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, método get y set de cada atributo excepto para la conexión, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.

Estas clases están vinculadas al subsistema de Proyectos.

* GestorUsuario.
  + Tendrá el método: Constructor, obtener todos los usuarios.
* Usuario
  + Tendrá los atributos nombre, correo y conexión.
  + Tendrá los métodos: Constructor, obtener todos los usuarios.

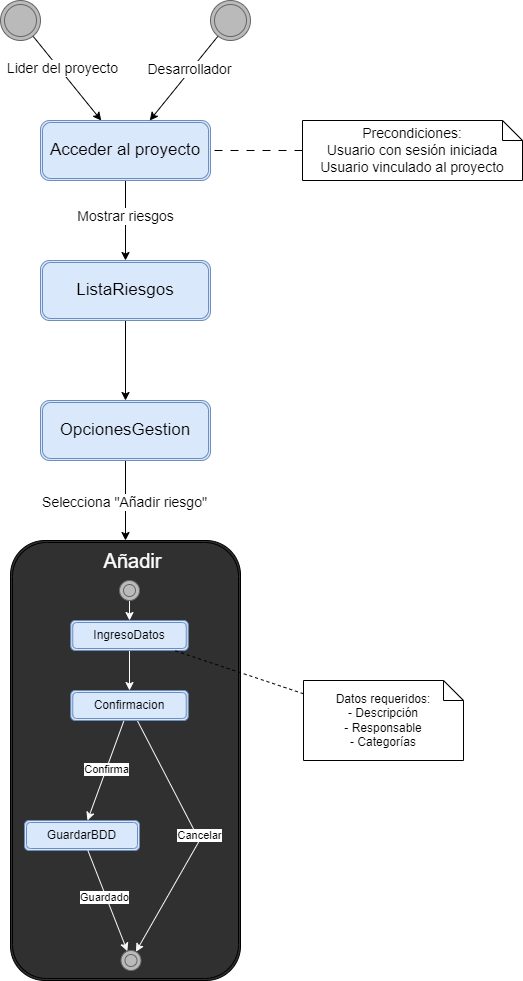
Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción

Diagrama, Esquemático

Descripción generada automáticamente

**Fig. 1.6**: Diagrama de secuencia CU04.



**Fig. 1.7**: Diagrama de flujo CU04.

Caso de Uso 5: Modificar lista de riesgos

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°5** | **Modificar lista de riesgos** | |
| Actor Principal | Líder del proyecto | |
| Precondiciones | 1. El usuario realizó el Caso de uso 1. 2. El usuario está vinculado a un proyecto. 3. Existe al menos un riesgo cargado. | |
| Disparador | El usuario selecciona la opción "Modificar lista de riesgos". | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona la opción "Modificar lista de riesgos". |
|  | 2 | La InterfazUsuario solicita *mostrarFormularioEdicion* al GestorRiesgos. |
|  | 3 | El GestorRiesgos envía el evento *obtenerDatosRiesgo* a Riesgo. |
|  | 4 | Riesgo solicita *obtenerDatosRiesgo* a la BDD. La BDD realiza la transacción y retorna los datos al Riesgo, el Riesgo retorna la respuesta al GestorRiesgos. |
|  | 5 | El GestorRiesgos retorna el formulario a la InterfazUsuario. |
|  | 6 | El actor modifica los datos permitidos (responsable y categoría) y selecciona "Guardar" (6a). Si selecciona "Cancelar", continúa con la extensión (6b). |
|  | 7 | La InterfazUsuario solicita *actualizarRiesgo(id\_riesgo, datos)* al GestorRiesgos. El GestorRiesgos valida los datos y envía el evento *actualizarRiesgo(id\_riesgo, datos)* a Riesgo. Riesgo solicita *actualizarRiesgo(id\_riesgo, datos)* a la BDD. La BDD actualiza el riesgo y retorna la respuesta al Riesgo, el Riesgo retorna la respuesta al GestorRiesgos. |
|  | 8 | El GestorRiesgos envía la confirmación a la InterfazUsuario y esta muestra un mensaje de éxito al actor. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 6a | 1. El GestorRiesgos detecta campos obligatorios sin completar. 2. El GestorRiesgos envía los errores a la InterfazUsuario. 3. La InterfazUsuario muestra los mensajes de error al actor. 4. El flujo continúa en el paso 3 de la descripción principal. |
|  | 6b | 1. La InterfazUsuario cierra el formulario de edición. 2. La InterfazUsuario retorna a la vista previa de la lista de riesgos. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.

Estas clases están vinculadas al subsistema de Riesgos.

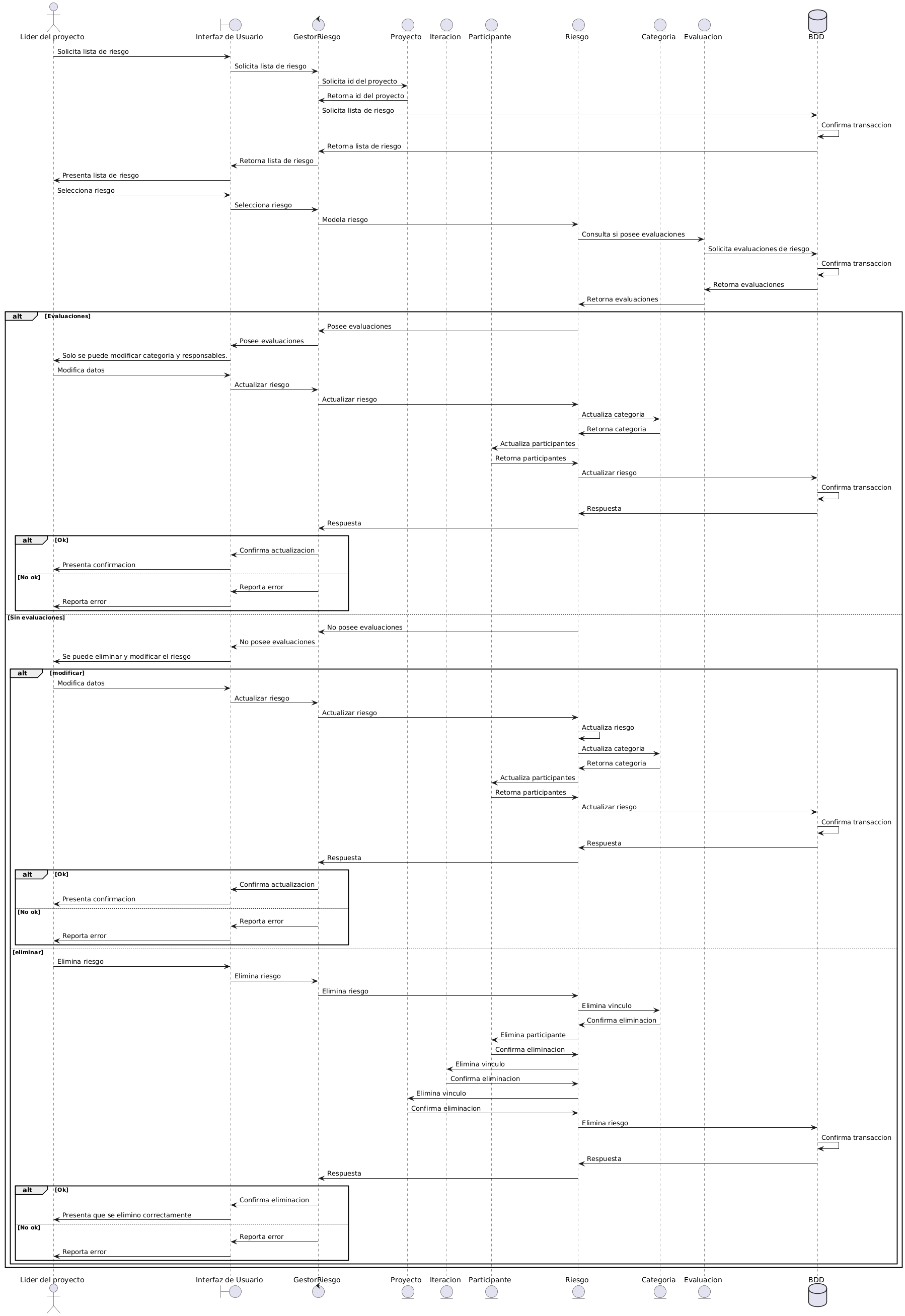
* GestorProyecto.
  + Tendrá los métodos: Constructor, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto, vincular o desvincular iteraciones, obtener todos los proyectos, vincular o desvincular participantes, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.
  + Comentario: Cada método deberá comprobar que no se pase ningún campo vacío, ni ninguna inyección sql.
* Proyecto.
  + Tendrá los atributos de nombre, descripción, estado, fecha inicio, fecha fin, iteraciones, participantes y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, retornar un proyecto por id, retornar proyecto por nombre, crear proyecto, actualizar proyecto, actualizar iteraciones, obtener todos los proyectos, vincular o desvincular participantes.
* Iteracion.
  + Tendrá los atributos fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, método get y set de cada atributo excepto para la conexión, obtener iteraciones por id de proyecto, actualizar iteración por id, eliminar iteración por id, añadir iteración.

Estas clases están vinculadas al subsistema de Proyectos.

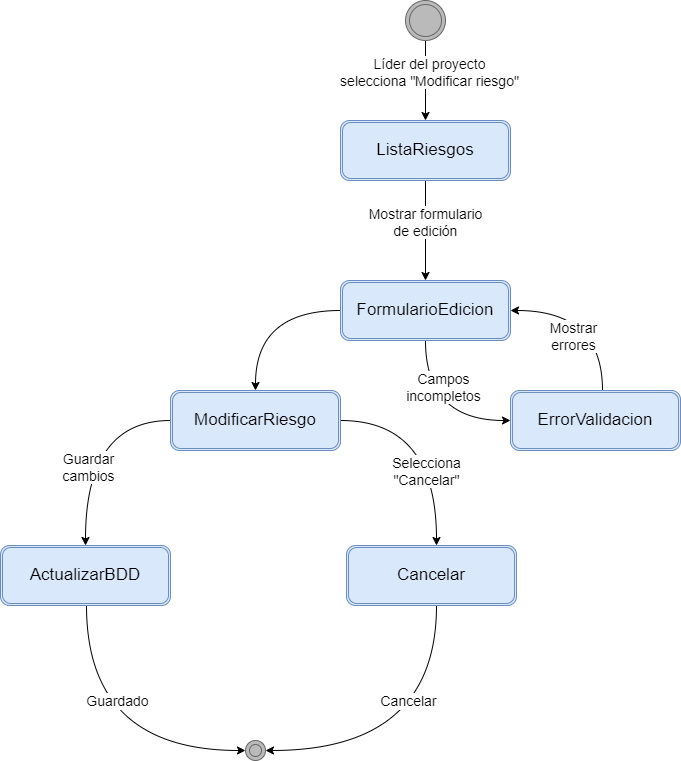
* GestorUsuario.
  + Tendrá el método: Constructor, obtener todos los usuarios.
* Usuario
  + Tendrá los atributos nombre, correo y conexión.
  + Tendrá los métodos: Constructor, obtener todos los usuarios.

Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción



**Fig. 1.8**: Diagrama de secuencia CU05.



**Fig. 1.9**: Diagrama de flujo CU05.

Caso de Uso 6: Administrar categorías de riesgos

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°6** | **Administrar categorías de riesgos** | |
| Actor Principal | Administrador del proyecto | |
| Precondiciones | 1. El usuario realizó el Caso de uso 1. 2. El usuario está vinculado a un proyecto. | |
| Disparador | El usuario selecciona la opción "Administrar categorías de riesgos". | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El usuario selecciona la opción "Administrar categorías de riesgos". |
|  | 2 | La InterfazUsuario solicita *mostrarListaCategorias* al GestorRiesgos. |
|  | 3 | El GestorRiesgos envía el evento *obtenerCategorias* a Categoria. |
|  | 4 | Categoria solicita *obtenerCategorias* a la BDD. La BDD realiza la transacción y retorna los datos al Categoria, el Categoria retorna la respuesta al GestorRiesgos. |
|  | 5 | El GestorRiesgos retorna la lista de categorías a la InterfazUsuario. |
|  | 6 | Si el usuario selecciona la opción "Crear nueva categoría" (6a); si el usuario selecciona la opción "Modificar categoría existente" (6b); si el usuario selecciona la opción "Cancelar" (6c). |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 6a | 1. La InterfazUsuario solicita *mostrarFormularioCreacionCategoria* al GestorRiesgos. 2. El usuario ingresa los datos, nombre y descripción, solicitados y selecciona la opción "Guardar". De lo contrario, presiona el botón "Cancelar" (6c). 3. La InterfazUsuario solicita *crearCategoria(datos)* al GestorRiesgos. El GestorRiesgos valida los datos y envía el evento *crearCategoria(datos)* a Categoria. Categoria solicita *crearCategoria(datos)* a la BDD. La BDD crea la categoría y retorna la respuesta al Categoria, el Categoria retorna la respuesta al GestorRiesgos. 4. El GestorRiesgos envía la confirmación a la InterfazUsuario y esta muestra un mensaje de éxito al usuario. 5. El caso de uso termina. |
|  | 6b | 1. La InterfazUsuario solicita *mostrarFormularioEdicionCategoria(id\_categoria)* al GestorRiesgos. El GestorRiesgos envía el evento *obtenerDatosCategoria(id\_categoria)* a Categoria. Categoria solicita *obtenerCategoria(id\_categoria)* a la BDD, la BDD retorna los datos al Categoria, este los retorna al GestorRiesgos. 2. El usuario ingresa los datos solicitados y selecciona la opción "Guardar". De lo contrario, presiona el botón "Cancelar" (6c). 3. La InterfazUsuario solicita *actualizarCategoria(id\_categoria, datos)* al GestorRiesgos. El GestorRiesgos valida los datos y envía el evento *actualizarCategoria(id\_categoria, datos)* a Categoria. Categoria solicita *actualizarCategoria(id\_categoria, datos)* a la BDD. La BDD actualiza la categoría y retorna la respuesta al Categoria, el Categoria retorna la respuesta al GestorRiesgos. 4. El GestorRiesgos envía la confirmación a la InterfazUsuario y esta muestra un mensaje de éxito al usuario. 5. El caso de uso termina. |
|  | 6c | 1. Se cancela la operación y se cierra el mensaje. 2. El caso de uso termina. |

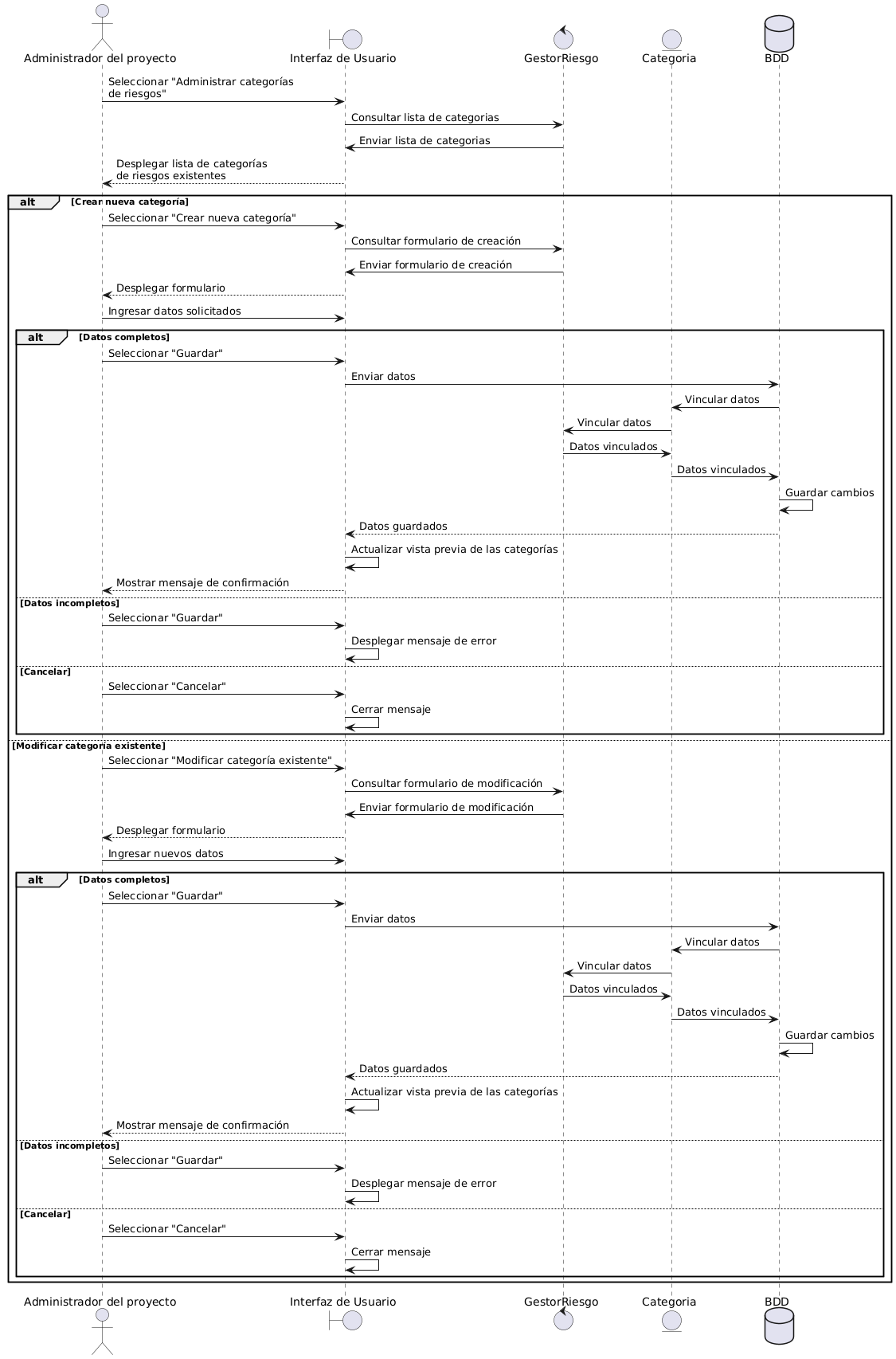
Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

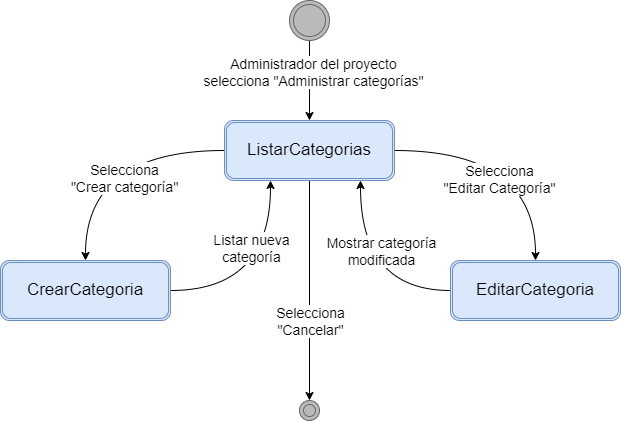
* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.

Estas clases están vinculadas al subsistema de Riesgos.

Diagrama de Interacción



**Fig. 1.10:** Diagrama de secuencia CU06.



**Fig. 1.11:** Diagrama de flujo CU06.

Caso de Uso 7: Realizar evaluación de riesgo

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°7** | **Realizar evaluación de riesgo** | |
| Actor Principal | Líder del proyecto; Desarrollador | |
| Precondiciones | * El usuario completó el Caso de uso 1. * El usuario está vinculado a un proyecto. * Existe al menos un riesgo cargado. | |
| Disparador | El usuario selecciona la opción "Realizar evaluación" en la lista de riesgos. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona la opción "Realizar evaluación" en la lista de riesgos. |
|  | 2 | La InterfazRiesgos solicita *listarRiesgos* al GestorRiesgos. |
|  | 3 | El GestorRiesgos envía el evento *listarRiesgos* a Riesgo. |
|  | 4 | Riesgo solicita *listarRiesgos* a la BDD, la BDD realiza la transacción y retorna la respuesta a Riesgo, Riesgo retorna la respuesta al GestorRiesgos. |
|  | 5 | El GestorRiesgos retorna los riesgos a la InterfazRiesgos. La InterfazRiesgos muestra la lista de riesgos al actor. |
|  | 6 | El actor selecciona el riesgo que desea evaluar. |
|  | 7 | El actor selecciona el riesgo que desea evaluar y selecciona "Evaluar" (7a) o "Cancelar" (7b). |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 7a | 1. La InterfazRiesgos solicita actualizarRiesgo(datos) al GestorRiesgos. 2. El GestorRiesgos valida los datos (7a1) y envía el evento actualizarRiesgo(datos) a Riesgo. 3. Riesgo solicita actualizarRiesgo(datos) a la BDD. La BDD actualiza el factor de riesgo. 4. La BDD retorna la respuesta a Riesgo, Riesgo retorna la respuesta al GestorRiesgos. 5. El GestorRiesgos retorna la respuesta a la InterfazRiesgos. 6. La InterfazRiesgos redirige al Caso de uso 12. 7. La InterfazRiesgos muestra un mensaje de confirmación de que la evaluación ha sido añadida exitosamente. |
|  | 7a1 | 1. Si el usuario no rellenó correctamente los datos obligatorios del campo, el GestorRiesgos retorna los errores. 2. La InterfazRiesgos muestra los mensajes indicando los errores cometidos por el usuario. 3. Vuelve al paso 7. |
|  | 7b | 1. La InterfazRiesgos cancela la operación y cierra el mensaje. |

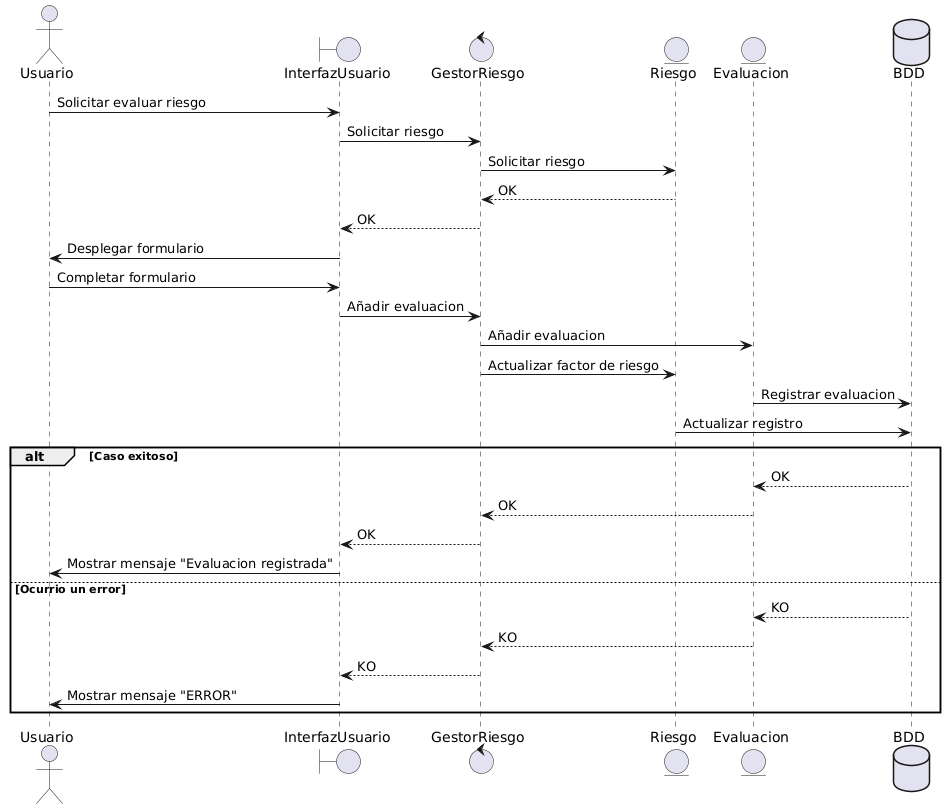
Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

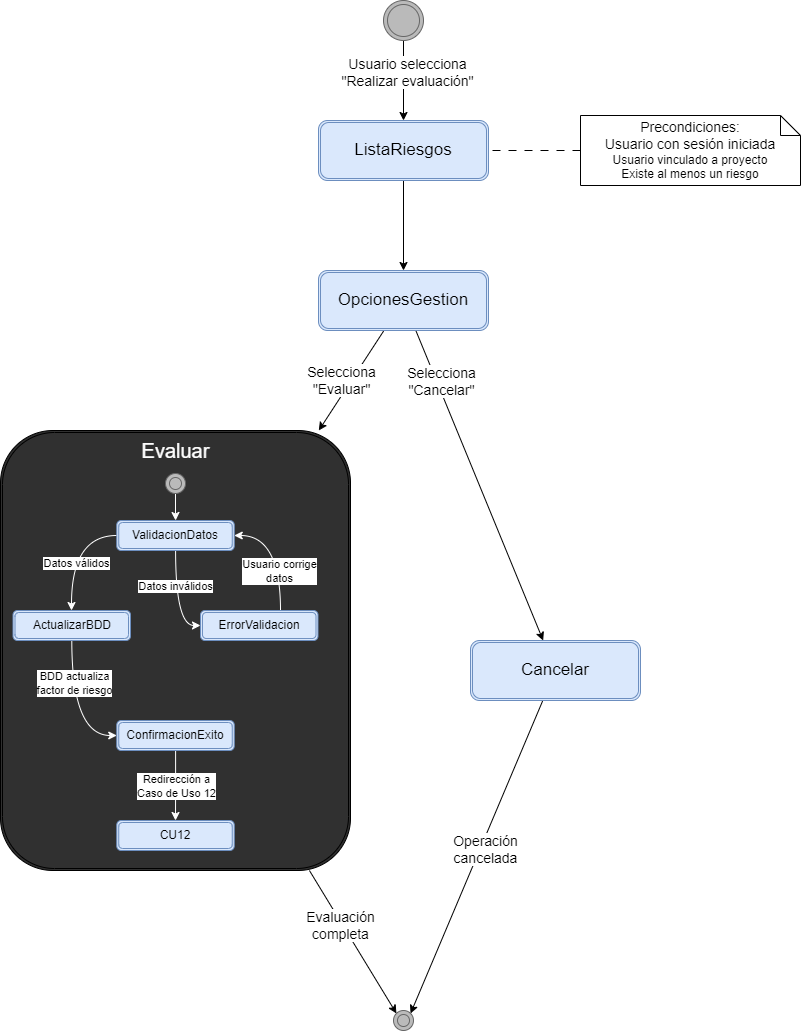
* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.

Estas clases están vinculadas al subsistema de Riesgos.

Diagrama de Interacción



**Fig. 1.12**: Diagrama de secuencia CU07.



**Fig. 1.13**: Diagrama de secuencia CU07.

Caso de Uso 8: Añadir plan de riesgo

Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°8** | **Añadir plan de riesgo** | |
| Actor Principal | Líder del proyecto; Desarrollador | |
| Precondiciones | * El actor completó el Caso de uso 1. * El actor está vinculado a un proyecto. * Existe al menos un riesgo cargado. | |
| Disparador | El usuario selecciona la opción "Realizar plan de riesgo" en la lista de riesgos. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor solicita añadirPlan(idRiesgo) a la InterfazUsuario. La InterfazUsuario redirige al usuario a la página /proyecto/:nombre/riesgos/agregarPlan. |
|  | 2 | El actor rellena el formulario de la página con los datos del plan de riesgo (id, tipo, descripción, tareas) (Excepción 2a) (Extensión 2b). |
|  | 3 | La InterfazUsuario solitica añadirPlan(idRiesgo, datosPlan) a GestorRiesgo. GestorRiesgo envia el evento añadirPlan(datosPlan) a Riesgo. Riesgo solicita actualizarRiesgo() a la BDD. La InterfazUsuario muestra un mensaje al usuario confirmando que se añadió el plan de riesgo. El actor selecciona confirmar. La InterfazUsuario redirige al actor a /proyecto/:nombre/riesgos |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 2a | El actor completo el formulario con campos inválidos (ninguna tarea agregada (debe haber al menos una), campos vacíos o tipo de plan invalido). La InterfazUsuario muestra un mensaje de error e indica los campos que deben ser corregidos. |
|  | 2b | El actor selecciona la opción “Agregar tarea” desde el formulario. La InterfazUsuario despliega un formulario para ingresar los datos de la tarea. El actor completa los datos. La InterfazUsuario solicita agregarTarea(datosTarea) al GestorRiesgo. El GestorRiesgo envia el evento añadirTareas() a PlanRiesgo. La InterfazUsuario devuelve al actor al formulario de plan de riesgos (Excepción 2bi). |
|  | 2bi | El actor completo el formulario de tarea con campos invalidos (nombre de longitud mayor a 30 caracteres, responsable inexistente o no participante del proyecto, campos vacíos, fechas de inicio-fin superpuestas, fechas de inicio-fin fuera de la iteración actual). La InterfazUsuario muestra un mensaje de error e indica los campos que deben ser corregidos. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

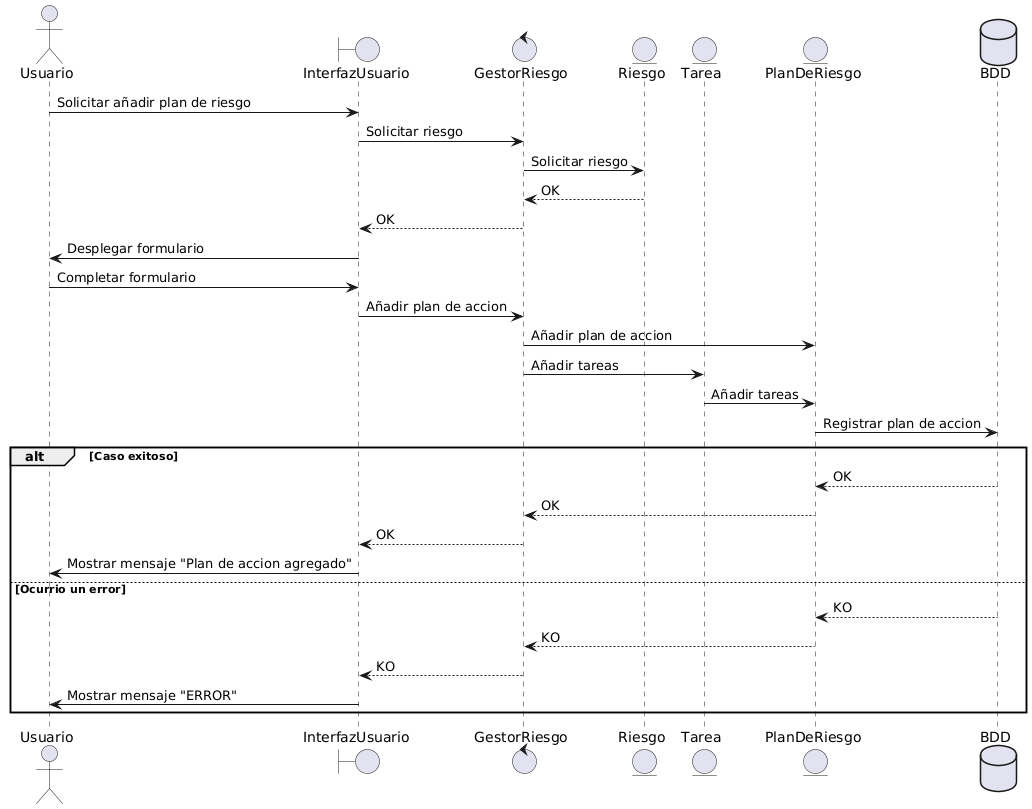
* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia
* PlanDeRiesgo
  + Tendrá los atributos: nombre, descripción, tipo, tareas y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Tareas:
  + Tendrá los atributos: nombre, descripción, estado, fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id.

Estas clases están vinculadas al subsistema de Riesgos.

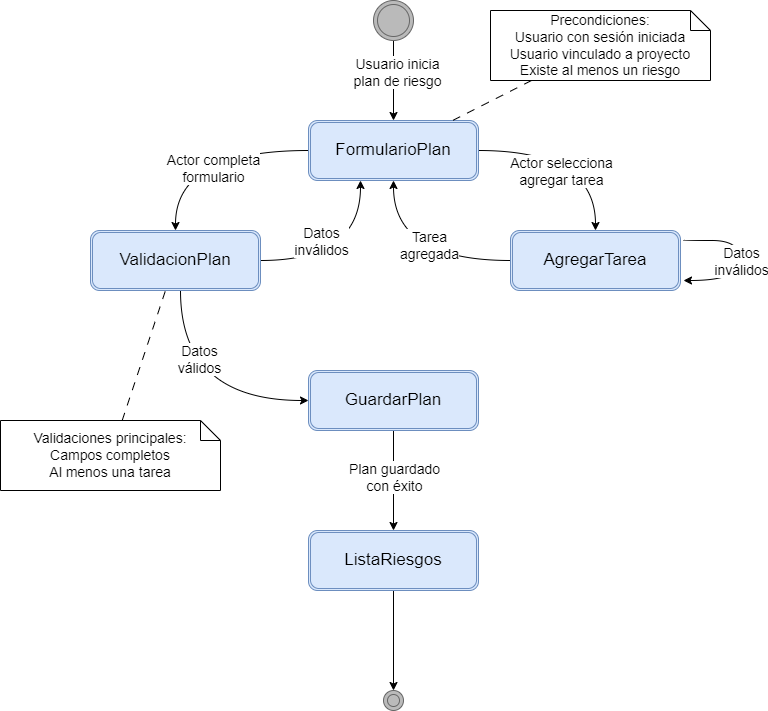
* GestorUsuario.
  + Tendrá el método: Constructor, obtener todos los usuarios.
* Usuario
  + Tendrá los atributos nombre, correo y conexión.
  + Tendrá los métodos: Constructor, obtener todos los usuarios.

Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción



**Fig. 1.14**: Diagrama de secuencia CU08.



**Fig. 1.15**: Diagrama de flujo CU08.

Caso de Uso 9: Modificar plan de acción

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°9** | **Modificar plan de acción** | |
| Actor Principal | Líder del proyecto | |
| Precondiciones | 1. El usuario realizó el Caso de uso 1. 2. El usuario está vinculado a un proyecto. 3. Existe al menos un riesgo cargado. | |
| Disparador | El usuario selecciona la opción "Modificar plan de acción" en la lista de riesgos. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona la opción "Modificar plan de acción" en la lista de riesgos. |
|  | 2 | La InterfazUsuario solicita *mostrarFormularioPlanAccion* al GestorPlanes. |
|  | 3 | El GestorPlanes envía el evento *obtenerDatosPlan* a PlanAccion. PlanAccion solicita *obtenerDatosPlan* a la BDD. La BDD realiza la transacción y retorna los datos al PlanAccion, el PlanAccion retorna la respuesta al GestorPlanes. El GestorPlanes retorna el formulario a la InterfazUsuario. |
|  | 4 | El actor modifica los datos del plan y selecciona "Guardar" (4a). Si selecciona "Cancelar", continúa con la extensión (4b). |
|  | 5 | La InterfazUsuario solicita *actualizarPlan(id\_plan, datos)* al GestorPlanes. El GestorPlanes valida los datos y envía el evento *actualizarPlan(id\_plan, datos)* a PlanAccion. PlanAccion solicita *actualizarPlan(id\_plan, datos)* a la BDD. La BDD actualiza el plan y retorna la respuesta al PlanAccion, el PlanAccion retorna la respuesta al GestorPlanes. |
|  | 6 | El GestorPlanes envía la confirmación a la InterfazUsuario y esta muestra un mensaje de éxito al actor. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 4a | 1. El GestorPlanes detecta campos obligatorios sin completar. 2. El GestorPlanes envía los errores a la InterfazUsuario. 3. La InterfazUsuario muestra los mensajes de error al actor. 4. El flujo continúa en el paso 4 de la descripción principal. |
|  | 4b | 1. La InterfazUsuario cierra el formulario de edición. 2. La InterfazUsuario retorna a la vista previa de la lista de riesgos. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

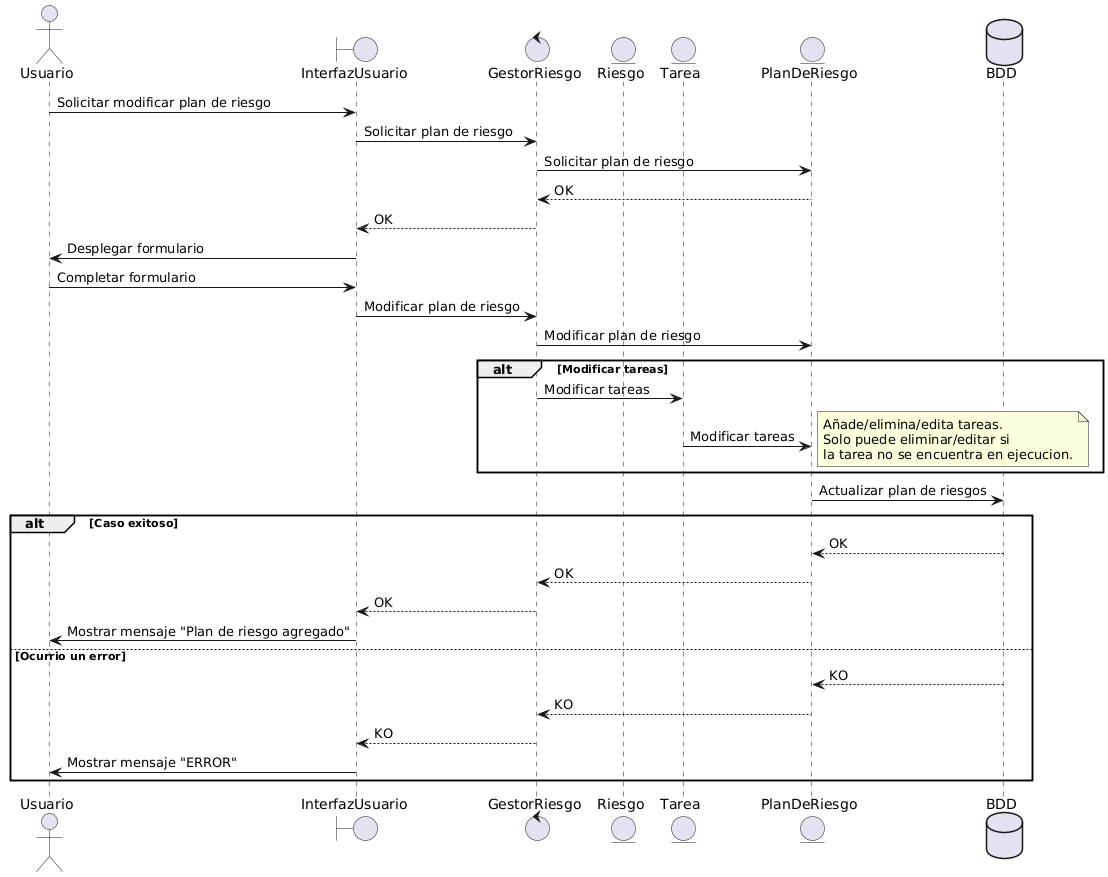
* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia
* PlanDeRiesgo
  + Tendrá los atributos: nombre, descripción, tipo, tareas y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Tareas:
  + Tendrá los atributos: nombre, descripción, estado, fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id.

Estas clases están vinculadas al subsistema de Riesgos.

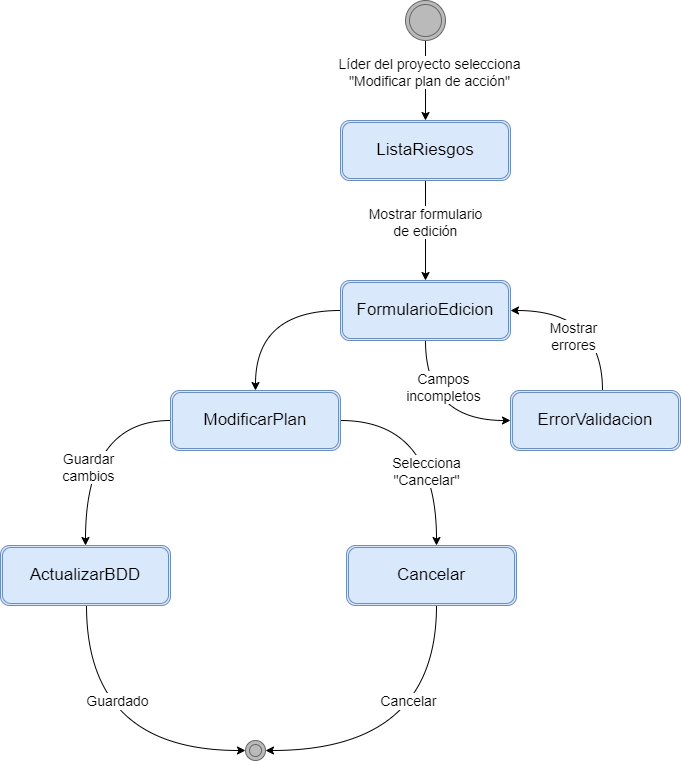
* GestorUsuario.
  + Tendrá el método: Constructor, obtener todos los usuarios.
* Usuario
  + Tendrá los atributos nombre, correo y conexión.
  + Tendrá los métodos: Constructor, obtener todos los usuarios.

Estas clases están vinculadas al subsistema de UARGflow.

Diagrama de Interacción



**Fig. 1.16**: Diagrama de secuencia CU09.



**Fig. 1.17**: Diagrama de secuencia CU09.

Caso de Uso 10: Realizar informes

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°10** | **Realizar informes** | |
| Actor Principal | Líder del proyecto, Desarrollador | |
| Precondiciones | 1. El usuario realizó el Caso de uso 1. 2. El usuario está vinculado a un proyecto. 3. Existe al menos un riesgo cargado. | |
| Disparador | El usuario selecciona la opción "Realizar informe" dentro del proyecto. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona el tipo de informe de incidencia. Si selecciona informe de tareas continua en 1a. Si selecciona informe de seguimiento continua en 1b. |
|  | 2 | La InterfazUsuario solicita desplegarFormularioIncidencia al GestorRiesgos. El GestorRiesgos envia el evento obtenerRiesgos a Riesgo. Riesgo solicita obtenerRiesgos a la BDD. |
|  | 3 | La InterfazUsuario despliega el formulario de incidencia. El actor ingresa los datos (Riesgo que ocurrió, fecha de la incidencia, daños producidos, descripción de la incidencia). |
|  | 4 | La InterfazUsuario solicita registrarIncidencia() al GestorRiesgos. El GestorRiesgos solicita registrarIncidencia() a Incidencia. Incidencia solicita registrarIncidencia() a la BDD. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 1a | 1. El actor selecciona informe de tareas. 2. La InterfazUsuario solicita generarInformeTareas() a GestorRiesgos. El GestorRiesgos envía el evento obtenerTareas a Tarea. El GestorRiesgos envía el evento generarInformeTareas() a Informe. |
|  | 1b | 1. El actor selecciona informe de seguimiento. 2. La InterfazUsuario solicita generarInformeSeguimiento() al GestorRiesgos. El GestorRiesgos envia el evento obtenerEvaluacion() a Evaluacion. El GestorRiesgos envia el evento obtenerPlanes() a PlanDeRiesgos. El GestorRiesgos envia el evento desplegarFormularioSeguimiento() a InterfazUsuario. La InterfazUsuario despliega el formulario de seguimiento de riesgos. 3. El actor ingresa los datos en el formulario (Estado del riesgo, resultado de los planes ejecutados). 4. La InterfazUsuario solicita registrarInformeSeguimiento() a GestorRiesgos. GestorRiesgos envia el evento registrarInformeSeguimiento() a Informe. Informe solicita registrarInformeSeguimiento() a la BDD. 5. La InterfazUsuario muestra el mensaje “Se ha registrado el informe de seguimiento correctamente” al actor. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

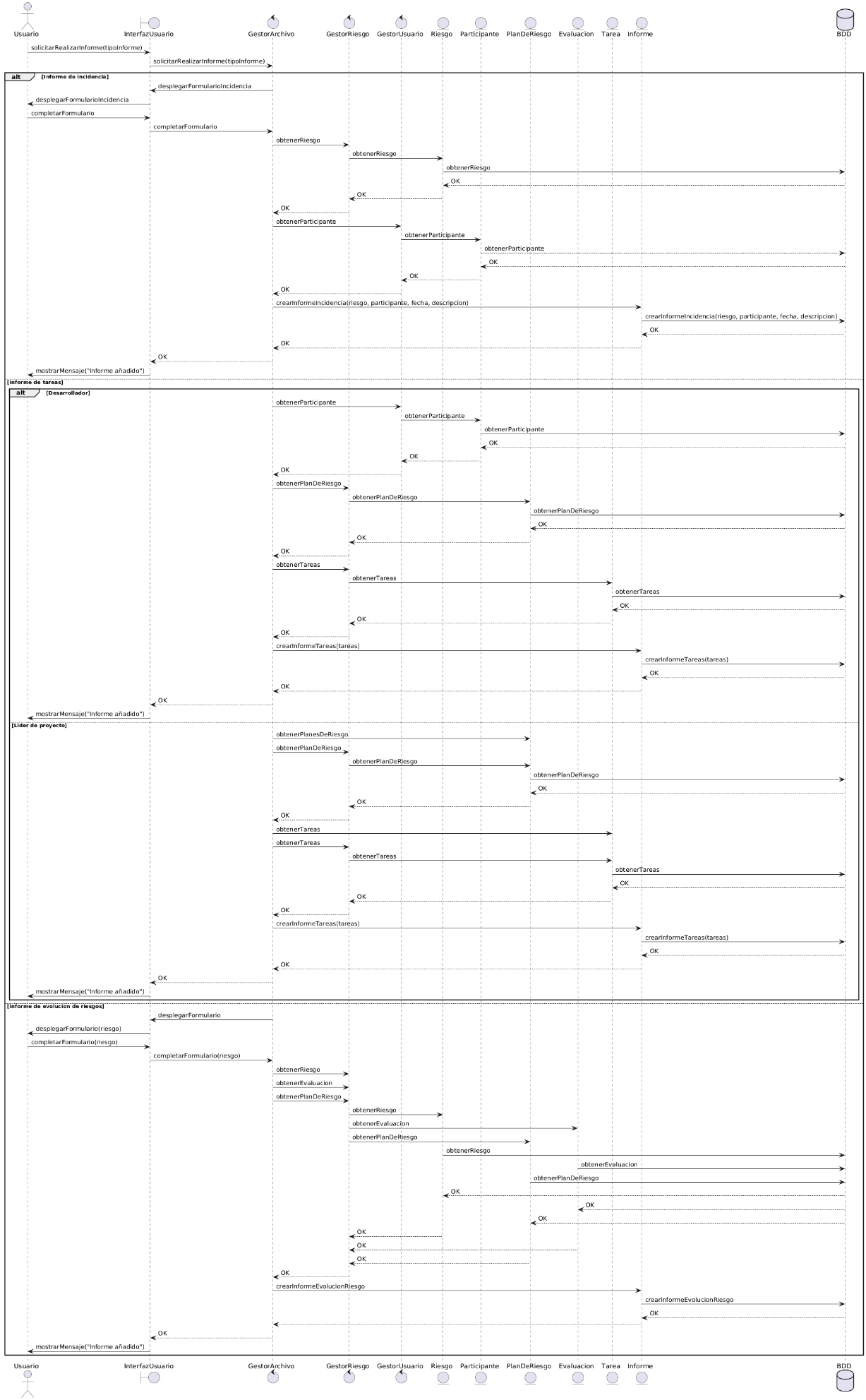
* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas, crear incidencia, obtener incidencias por id, obtener todas las incidencias.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia
* PlanDeRiesgo
  + Tendrá los atributos: nombre, descripción, tipo, tareas y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Tareas:
  + Tendrá los atributos: nombre, descripción, estado, fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id.
* Incidencia:
  + Tendrá los atributos: descripción y conexión.
  + Tendrá los métodos: Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear incidencia, obtener incidencias por id de riesgo, obtener todas las incidencias.
  + Comentario: Es un tipo de informe.

Estas clases están vinculadas al subsistema de Riesgos.

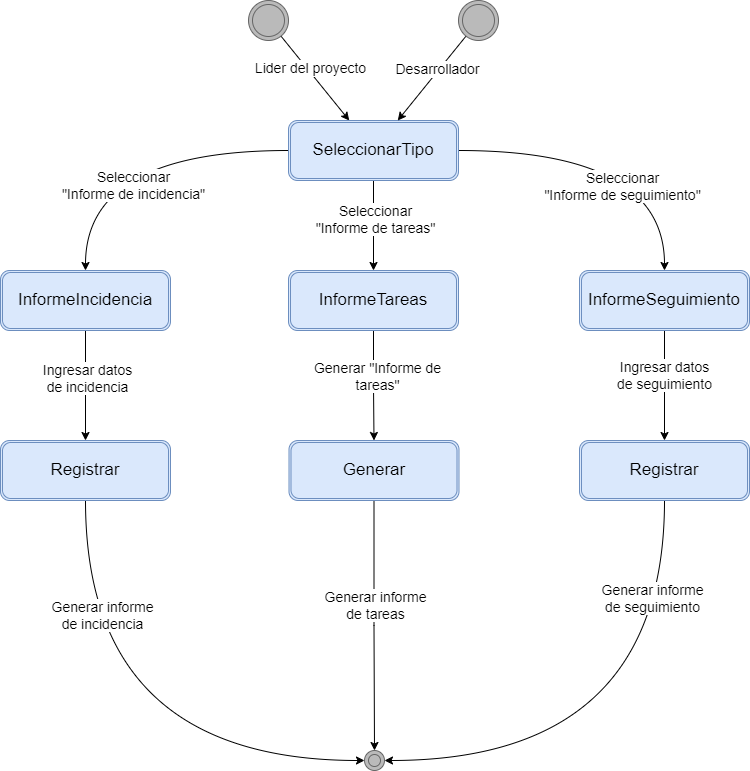
* GestorArchivos.
  + Tendrá los métodos: generar un informe, obtener informe de incidencia por id, descargar archivo como pdf, descargar archivo como xslx.
* Informe
  + Tendrá los atributos: datos.
  + Tendrá los métodos: Constructor, los métodos get y set de cada uno de los atributos, generar un informe, obtener informe de incidencia por id.
* Archivo
  + Tendrá el atributo: informe.
  + Tendrá los métodos: Constructor, método get y set del atributo informe, descargar archivo como pdf, descargar archivo como xslx.

Estas clases están vinculadas al subsistema de Exportar.

Diagrama de Interacción



**Fig. 1.18**: Diagrama de secuencia CU10.



**Fig. 1.19**: Diagrama de flujo CU10.

Caso de Uso 11: Exportar archivos

### Especificación de caso de uso de diseño

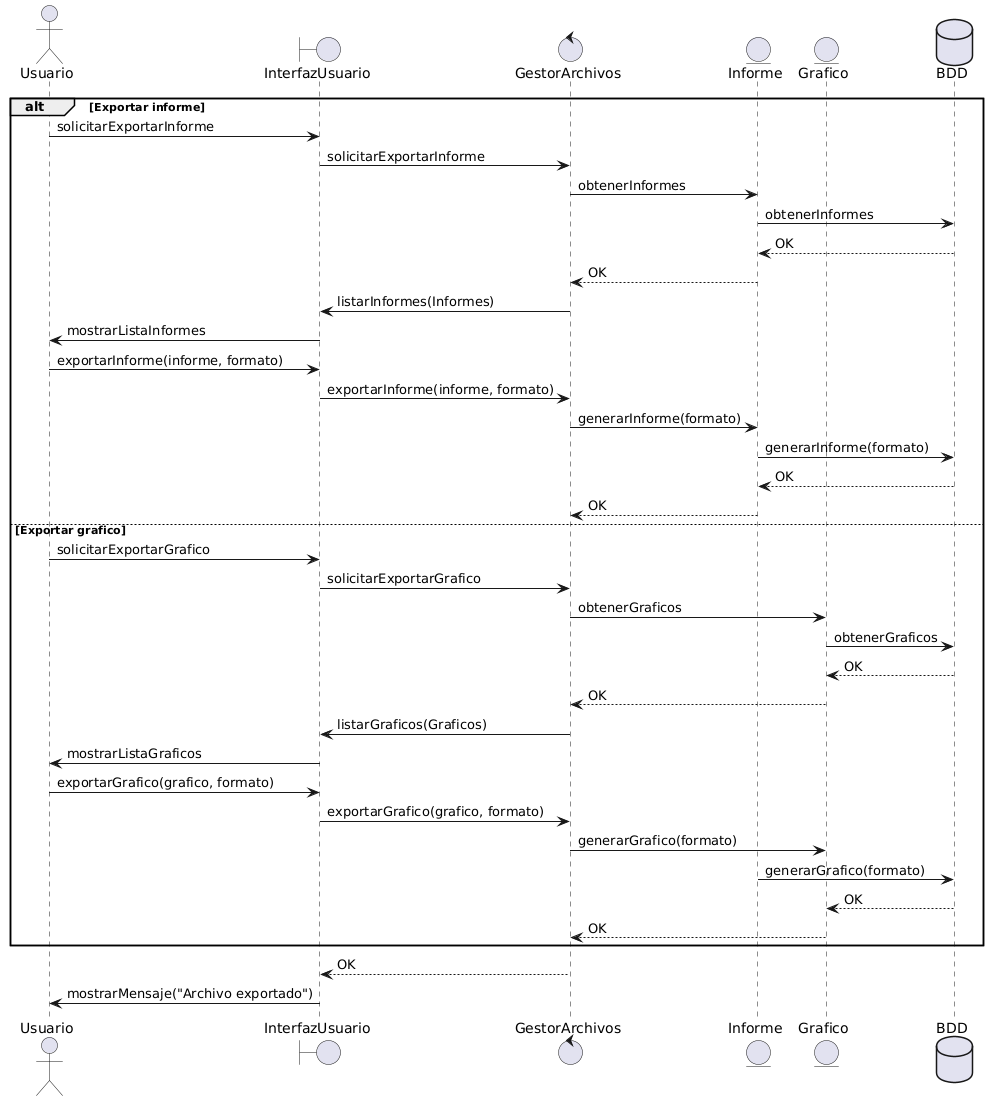
|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°11** | **Exportar archivos** | |
| Actor Principal | Líder de proyecto, Desarrollador | |
| Precondiciones | 1. El usuario realizó el Caso de uso 1. 2. El usuario está vinculado a un proyecto. | |
| Disparador | El actor selecciona la opción “Exportar” sobre un gráfico generado por el sistema o sobre un informe. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor selecciona exportar informe. Si intenta exportar un gráfico (1a). |
|  | 2 | La InterfazUsuario solicita obtenerInformes a GestorRiesgos. GestorRiesgos envía el evento obtenerInformes a Informe. Informe solicita obtenerInformes a la BDD. El GestorRiesgos envia el evento listarInformes a la InterfazUsuario. La InterfazUsuario lista los informes existentes o que pueden ser generados por el sistema (informe de tareas). 3 El actor selecciona el informe a exportar. La InterfazUsuario solicita exportarInforme a GestorRiesgos. GestorRiesgos envia el evento exportarInforme a Informe. |
|  | 3 | El actor selecciona el informe a exportar y la InterfazUsuario presenta las opciones de formato de exportación (PDF, Excel, CSV, etc.). El actor selecciona el formato deseado y confirma la exportación. Si el actor cancela la exportación, continúa con el Subflujo 1. |
|  | 4 | La InterfazUsuario solicita generarArchivoExportado a GestorRiesgos. GestorRiesgos envia el evento generarArchivoExportado a Informe. Informe genera el archivo en el formato seleccionado. |
|  | 5 | La InterfazUsuario muestra un mensaje de confirmación indicando que la exportación se realizó correctamente. |
|  | 6 | El caso de uso termina. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 1a | El actor selecciona exportar gráfico. La InterfazUsuario inicia la descarga del gráfico en formato .png. |

Diagrama de paquetes

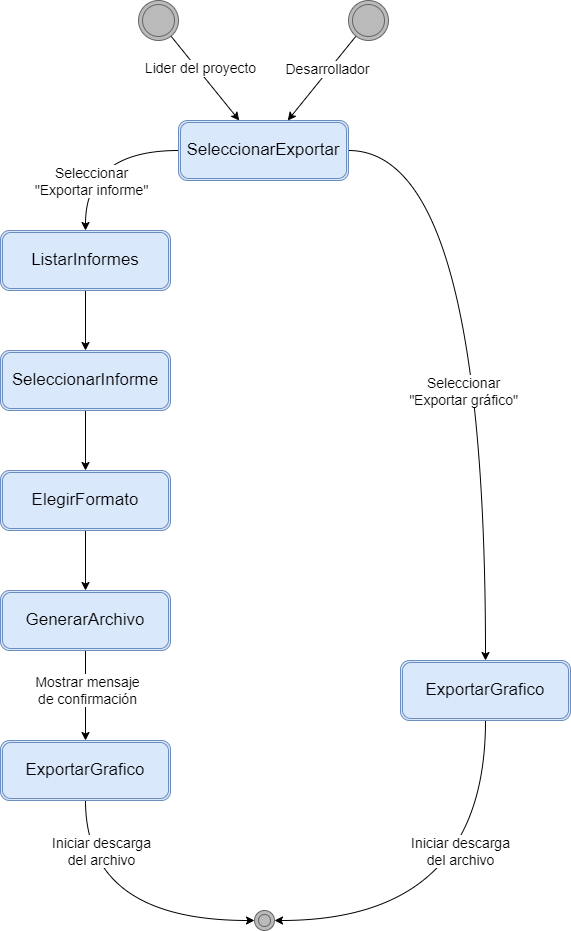
* GestorArchivos.
  + Tendrá los métodos: generar un informe, obtener informe de incidencia por id, descargar archivo como pdf, descargar archivo como xslx.
* Informe
  + Tendrá los atributos: datos.
  + Tendrá los métodos: Constructor, los métodos get y set de cada uno de los atributos, generar un informe, obtener informe de incidencia por id.
* Archivo
  + Tendrá el atributo: informe.
  + Tendrá los métodos: Constructor, método get y set del atributo informe, descargar archivo como pdf, descargar archivo como xslx.

Estas clases están vinculadas al subsistema de Exportar.

Diagrama de Interacción



**Fig. 1.20**: Diagrama de secuencia CU11.



**Fig. 1.21**: Diagrama de flujo CU11.

Caso de Uso 12: Realizar análisis de riesgo

### Especificación de caso de uso de diseño

|  |  |  |
| --- | --- | --- |
| **CASO DE USO N°12** | **Realizar análisis de riesgo** | |
| Actor Principal | Líder de proyecto, Desarrollador | |
| Precondiciones | Existe al menos un riesgo cargado. | |
| Disparador | Un actor añade un riesgo, evaluación o plan de mitigación o minimización y de contingencia, o un proyecto avanza a la siguiente iteración. | |
| DESCRIPCION | Paso | Acción |
|  | 1 | El actor añade un riesgo.  Si el actor añade una evaluación continua en extensión 1a.  Si el actor añade un plan de contingencia y de minimización o mitigación continua en extensión 1b.  Si el proyecto avanza a la siguiente iteración continua en extensión 1c. |
|  | 2 | El GestorRiesgo solicita a InterfazUsuario marcarRiesgo(“Evaluar”). La InterfazUsuario muestra una marca visual del riesgo, notificando que necesita ser evaluado. |
| EXTENSIONES | Paso | Acción de Derivación |
|  | 1a | Si el factor de riesgo resultado de la evaluación es menor a 9 el GestorRiesgo envía el evento marcarRiesgo(“Ignorar”) a InterfazUsuario. La InterfazUsuario muestra una marca visual del riesgo, notificando que no necesita acciones y deshabilita las opciones evaluar y planificar.  Si el factor de riesgo resultado de la evaluación es mayor igual a 9 y menor que 36 el GestorRiesgo envía el evento marcarRiesgo(“Reevaluar”) a InterfazUsuario. La InterfazUsuario muestra una marca visual del riesgo, notificando que necesita ser reevaluado en la próxima iteración, y deshabilita las opciones evaluar y planificar.  Si el factor de riesgo resultado de la evaluación es mayor o igual a 36 y menor que 64 el GestorRiesgo envia el evento marcarRiesgo(“Planificar”) a InterfazUsuario. La InterfazUsuario muestra una marca visual del riesgo, notificando que necesita ser planificado, y habilita la opción planificar.  Si el factor de riesgo resultado de la evaluación es mayor o igual que 64 el GestorRiesgo envía el evento marcarRiesgo(“Critico”) a InterfazUsuario. La InterfazUsuario muestra una marca visual del riesgo, notificando que necesita ser planificado de manera urgente, y habilita la opción planificar. |
|  | 1b | El GestorRiesgo envía el evento marcarRiesgo(“Planificado”) a InterfazUsuario. La InterfazUsuario muestra una marca visual del riesgo, notificando que existen planes de riesgo de ese riesgo para la iteración actual. |
|  | 1c | El GestorRiesgo envia el evento marcarRiesgos() a InterfazUsuario. La InterfazUsuario muestra una marca visual en todos los riesgos activos (que no estén marcados como “ignorar”), notificando que los riesgos necesitan ser evaluados, habilita la opción de evaluación y deshabilita la de planificación. |

Diagrama de paquetes

Los objetos que fueron identificados en este caso de uso fueron:

* GestorRiesgo.
  + Tendrá los métodos: Constructor, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, vincular o desvincular responsable, actualizar categoría de un riesgo, vincular o desvincular iteraciones, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas, crear incidencia, obtener incidencias por id, obtener todas las incidencias, comprobar si riesgo necesita evaluación, comprobar si riesgo necesita un plan, comprobar si riesgo necesita dos planes.
* Riesgo.
  + Tendrá los atributos: descripción, factor riesgo, categoría, iteración y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, agregar riesgo, obtener todos los riesgos de un proyecto, obtener riesgo por id, actualizar riesgo por id, eliminar riesgo por id, actualizar categoría de un riesgo, vincular o desvincular proyecto, vincular o desvincular iteraciones.
* Categoria
  + Tendrá los atributos: nombre, descripción y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Agregar categoría, actualizar categoría, eliminar categoría, obtener todas las categorías.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* Evaluacion
  + Tendrá los atributos: impacto, probabilidad y conexión
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear una evaluación, obtener evaluaciones de un riesgo, obtener todas las evaluaciones, actualizar evaluación por id, eliminar evaluación por id.
  + Comentario: Esta clase se utiliza de manera implícita en este diagrama de secuencia.
* PlanDeRiesgo
  + Tendrá los atributos: nombre, descripción, tipo, tareas y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear plan de riesgo, actualizar plan de riesgo por id, eliminar plan de riesgo por id, vincular iteración, Vincular o desvincular tareas.
* Tareas:
  + Tendrá los atributos: nombre, descripción, estado, fecha inicio, fecha fin y conexión.
  + Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, Crear tarea, modificar tarea por id, vincular o desvincular participante, eliminar tarea por id.
* Incidencia:
  + Tendrá los atributos: descripción y conexión
  + Tendrá los métodos: Tendrá los métodos: Constructor, los métodos get y set de cada atributo excepto para conexión, crear incidencia, obtener incidencias por id de riesgo, obtener todas las incidencias.
  + Comentario: Es un tipo de informe.

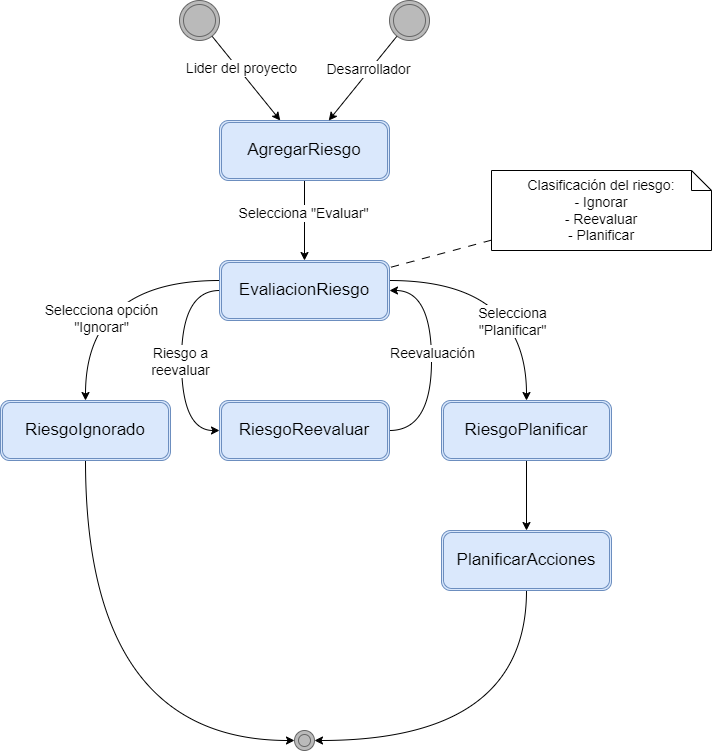
Estas clases están vinculadas al subsistema de Riesgos.

Diagrama de Interacción

Tabla

Descripción generada automáticamente

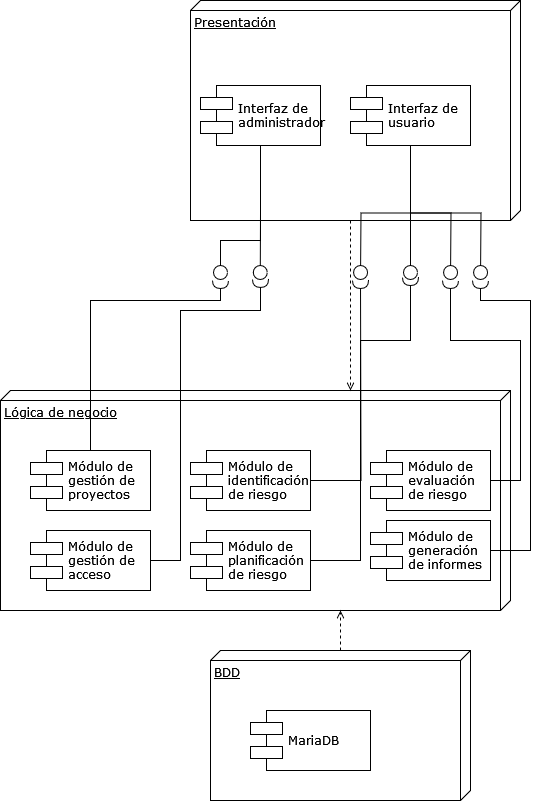
**Fig. 1.22**: Diagrama de secuencia CU12.



**Fig. 1.23**: Diagrama de flujo CU12.

# Diagramas

Diagrama de componentes



**Fig. 2.1**: Modelo de componentes.

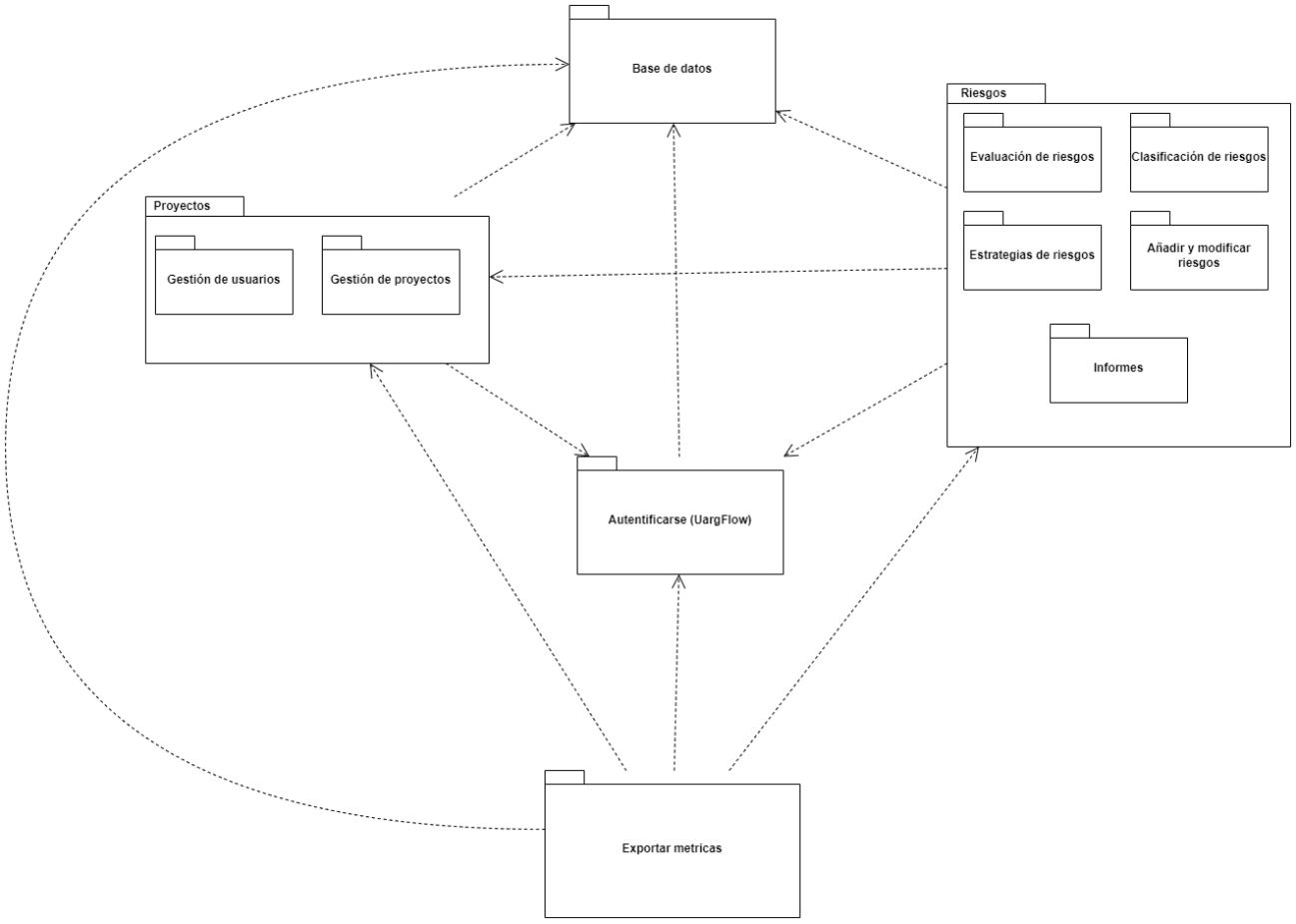
Diagrama de Clases



**Fig. 2.2**: Diagrama de Clases.

**Aclaracion**: Este diagrama esta pendiente a modificaciones.

Diagramas de Paquetes

  
**Fig. 2.3**: Diagrama de Paquetes.