Fase II

Proyecto página web de gestión de proyectos con SCRUM

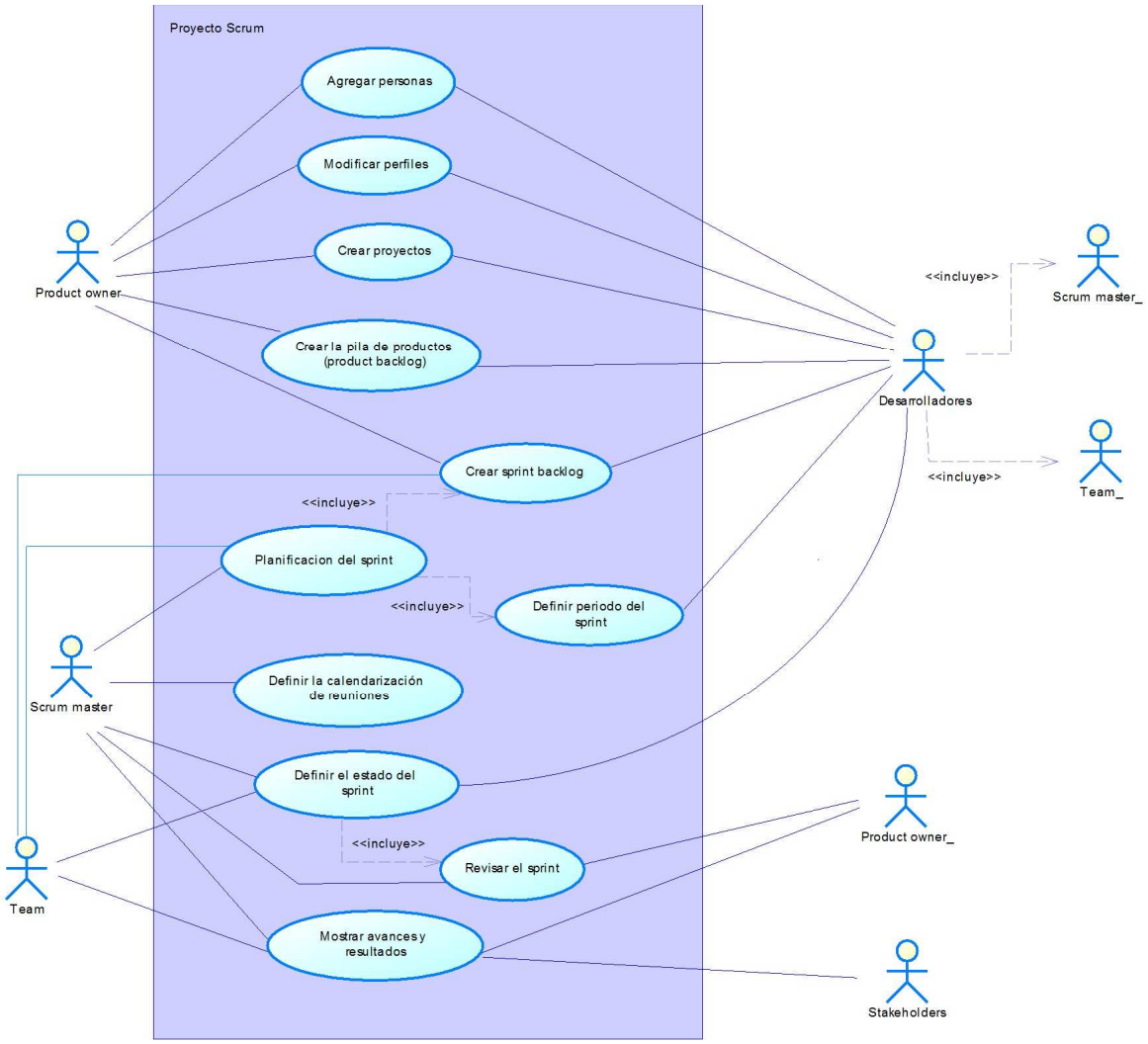
11/09/2012

Universidad Rafael Landívar – Campus Quetzaltenango
Linda Estrella Córdova Monterroso (1617009)
Ludwing Juan Homero Pérez Tzaquitzal (1520909)
Luis Eduardo Rivera Sánchez (1585509)
Mario Rolando Valdés Argueta (1543709)

Índice

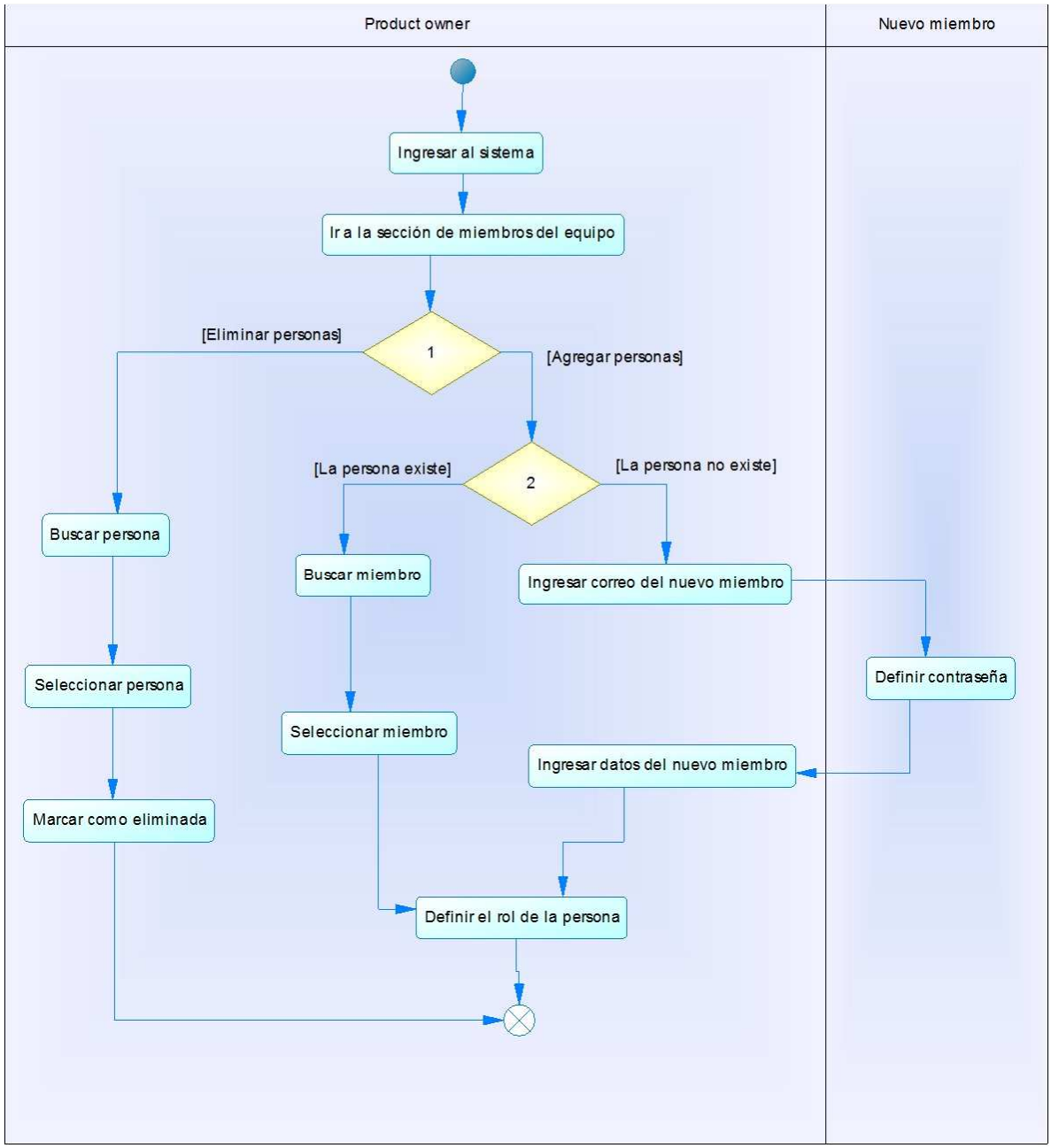
Diagramas

Diagrama de casos de uso

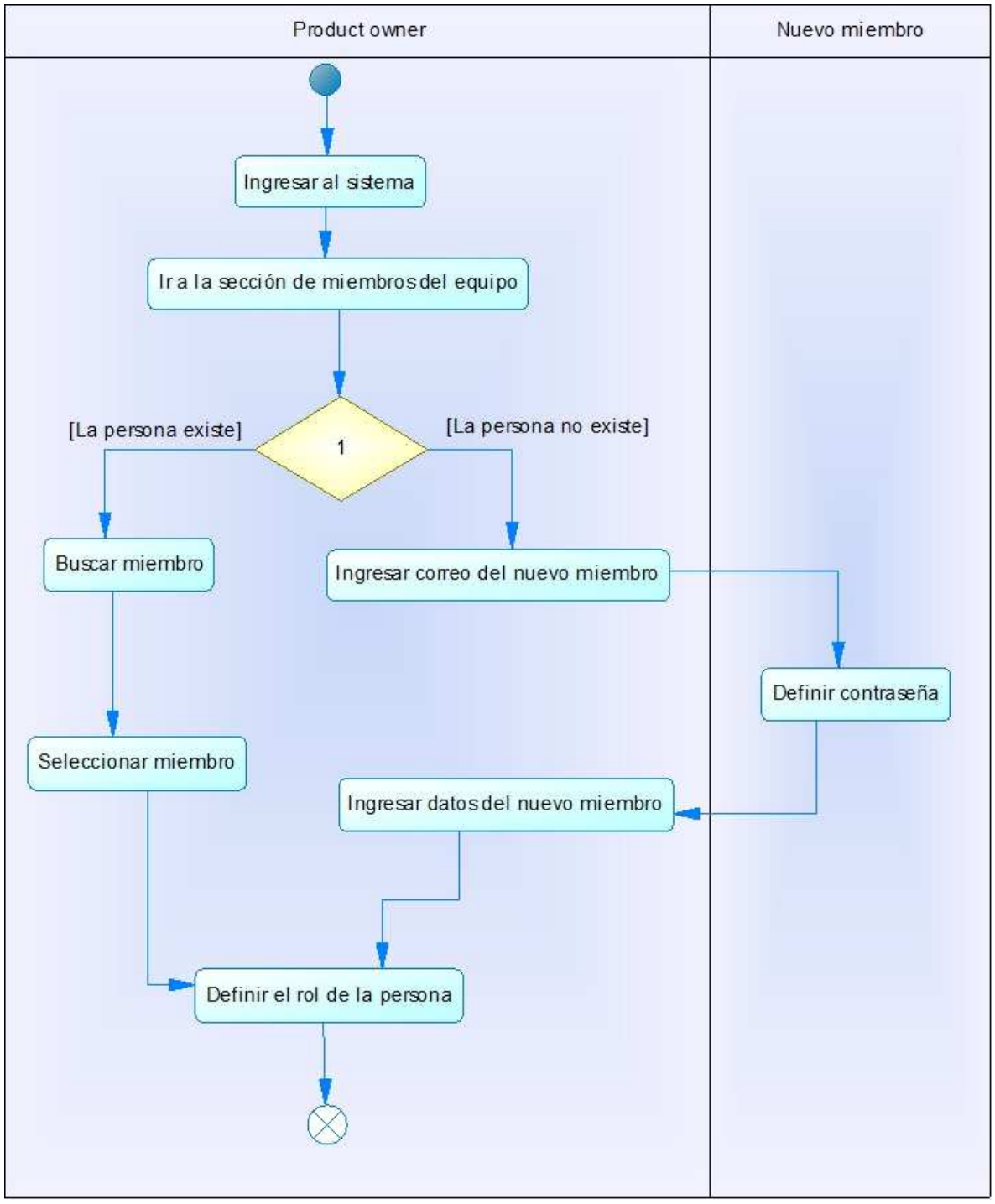


Diagramas de actividades

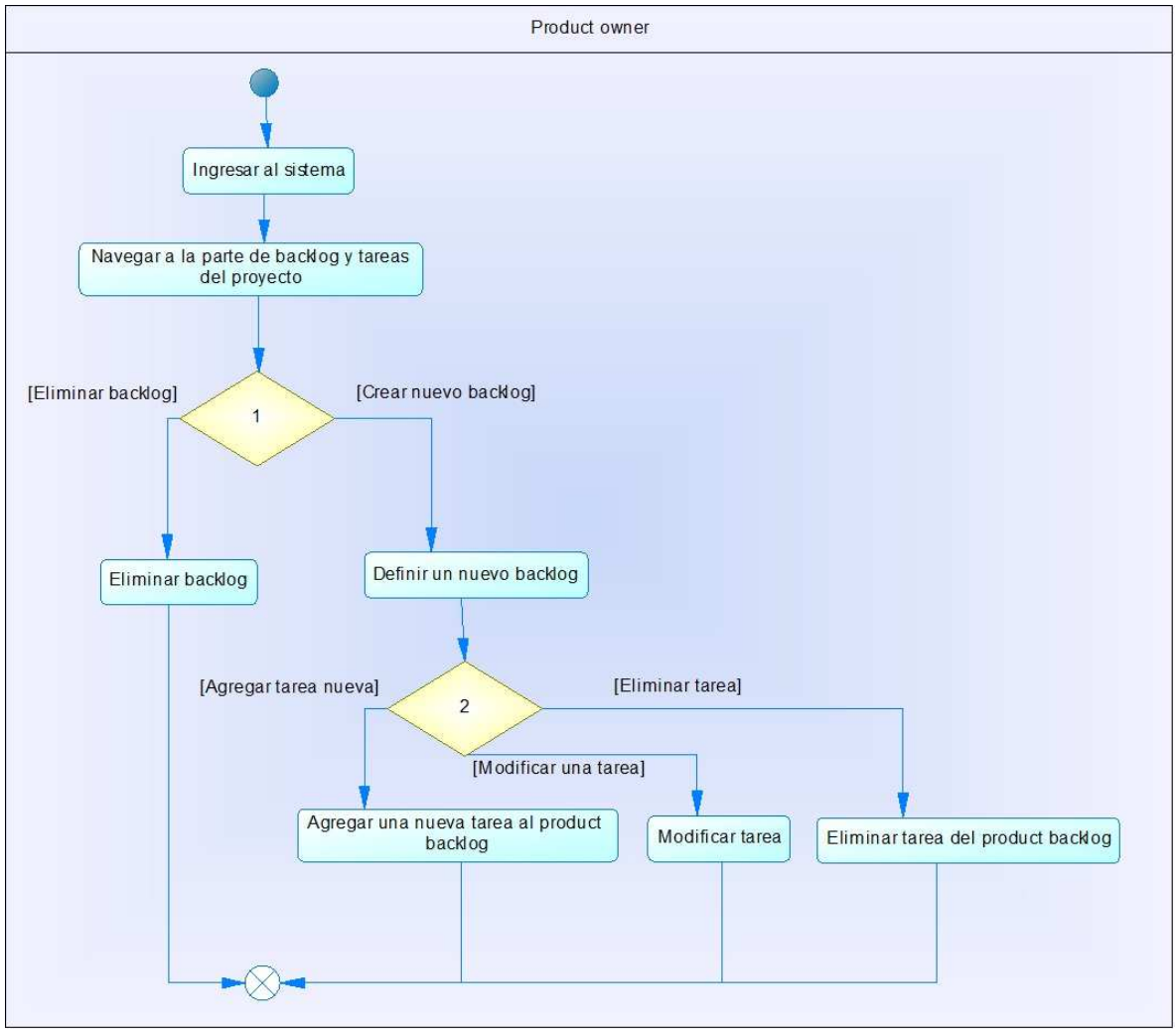
Diagrama de gestión de personas (miembros del equipo)



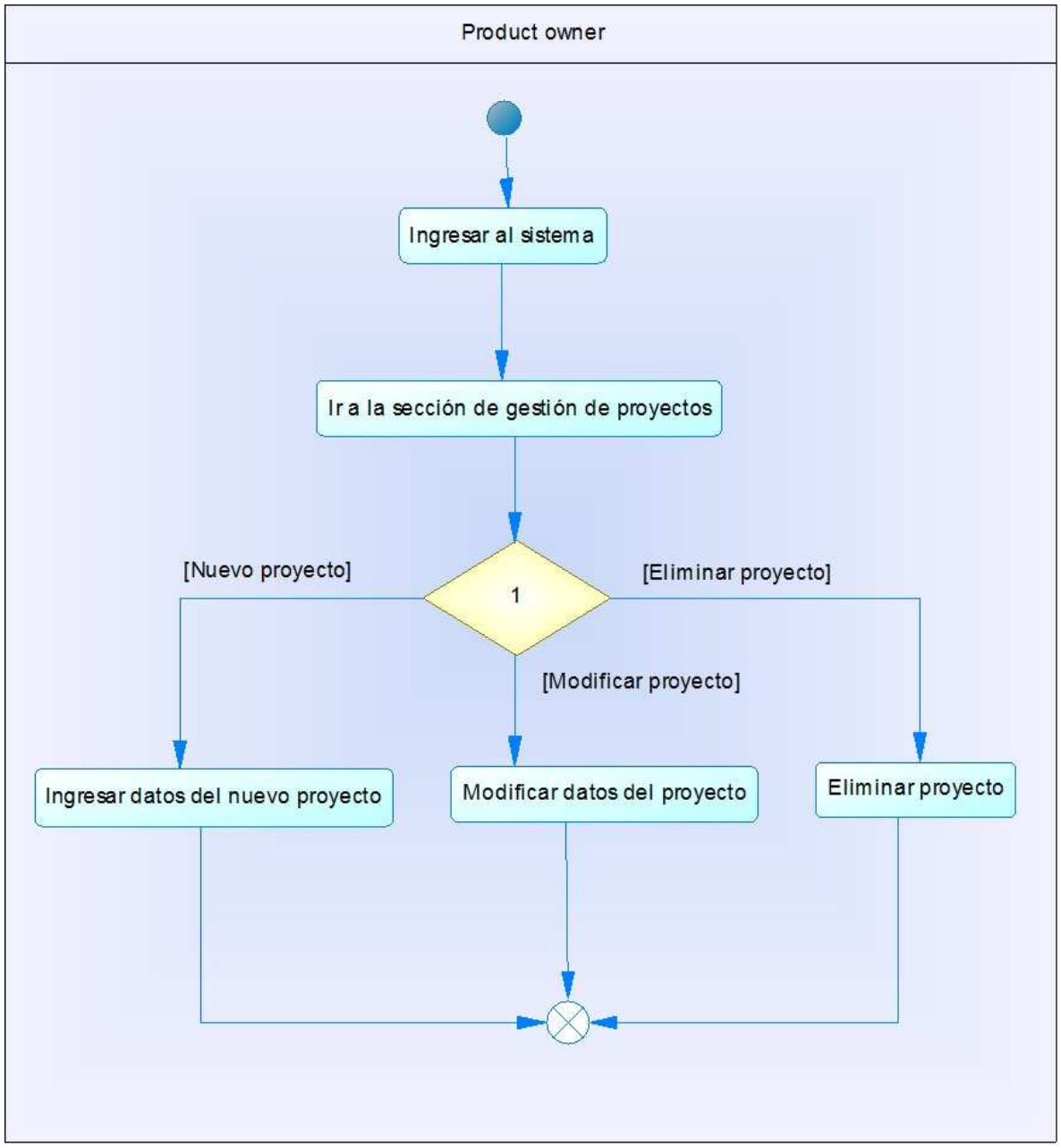
Agregar personas



Crear product backlog (listado de características)

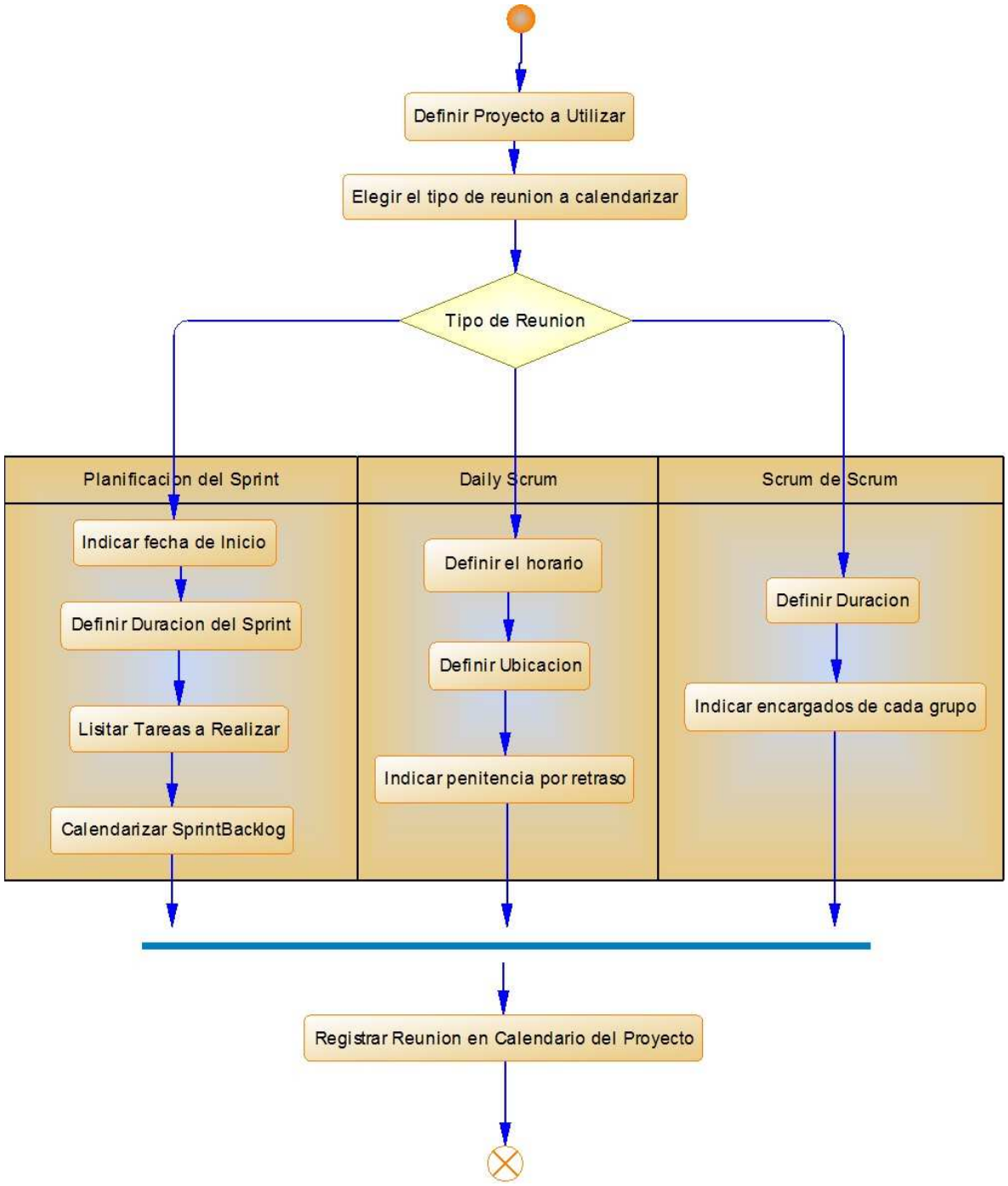


Gestión de proyectos

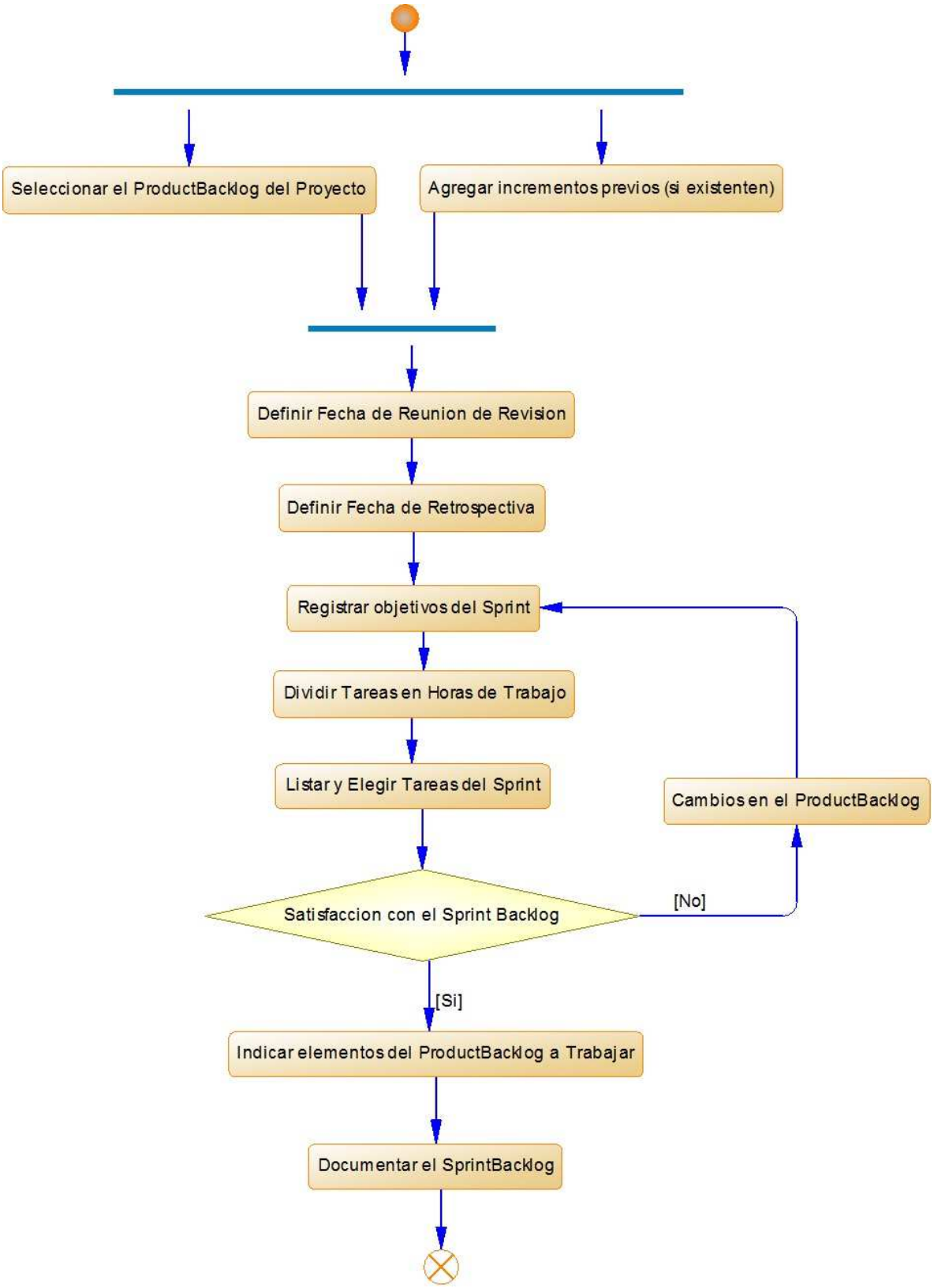


Crear nuevo proyecto

Gestión de reuniones:



Gestión de sprint Backlog



Gestión de sprints

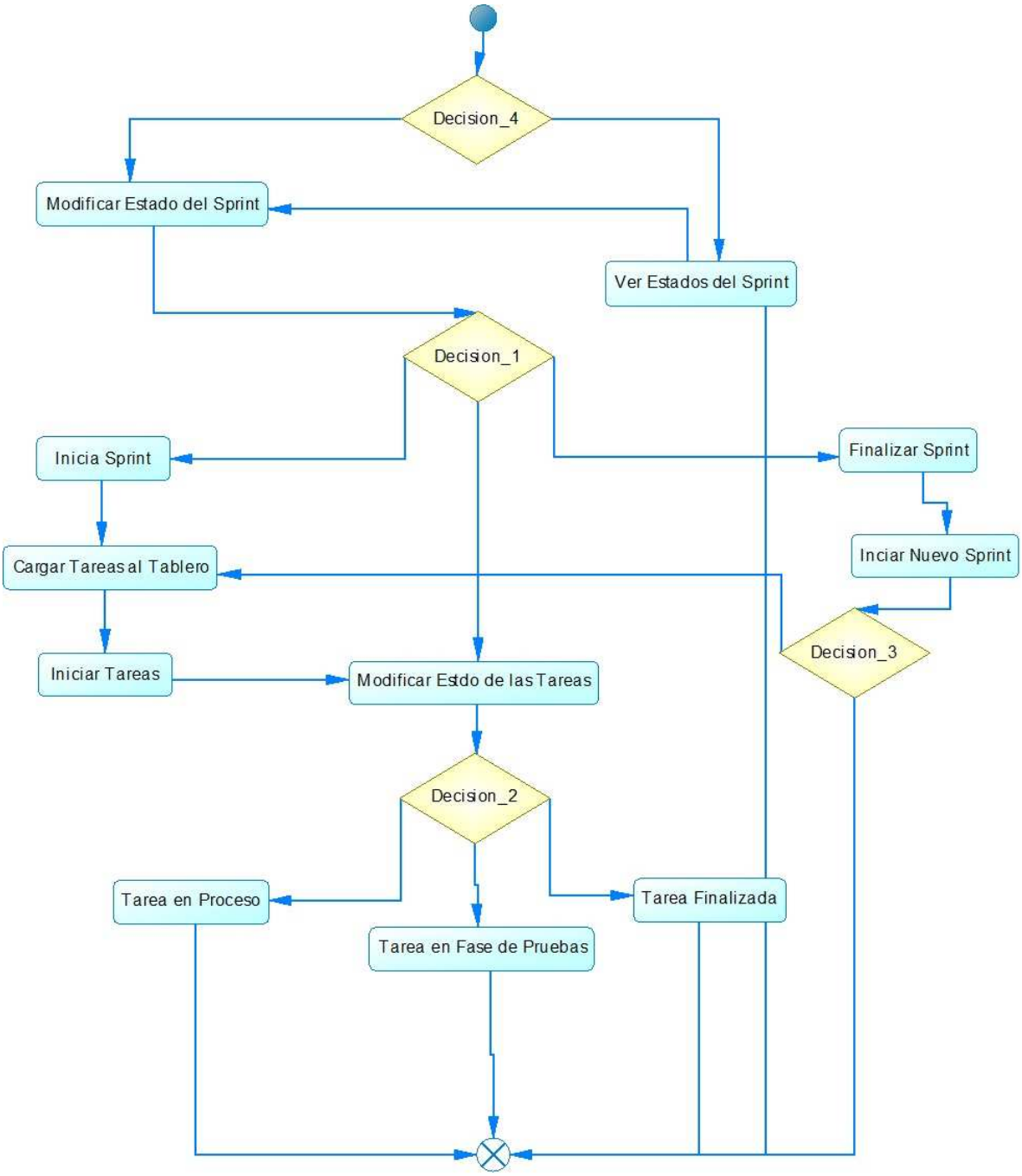
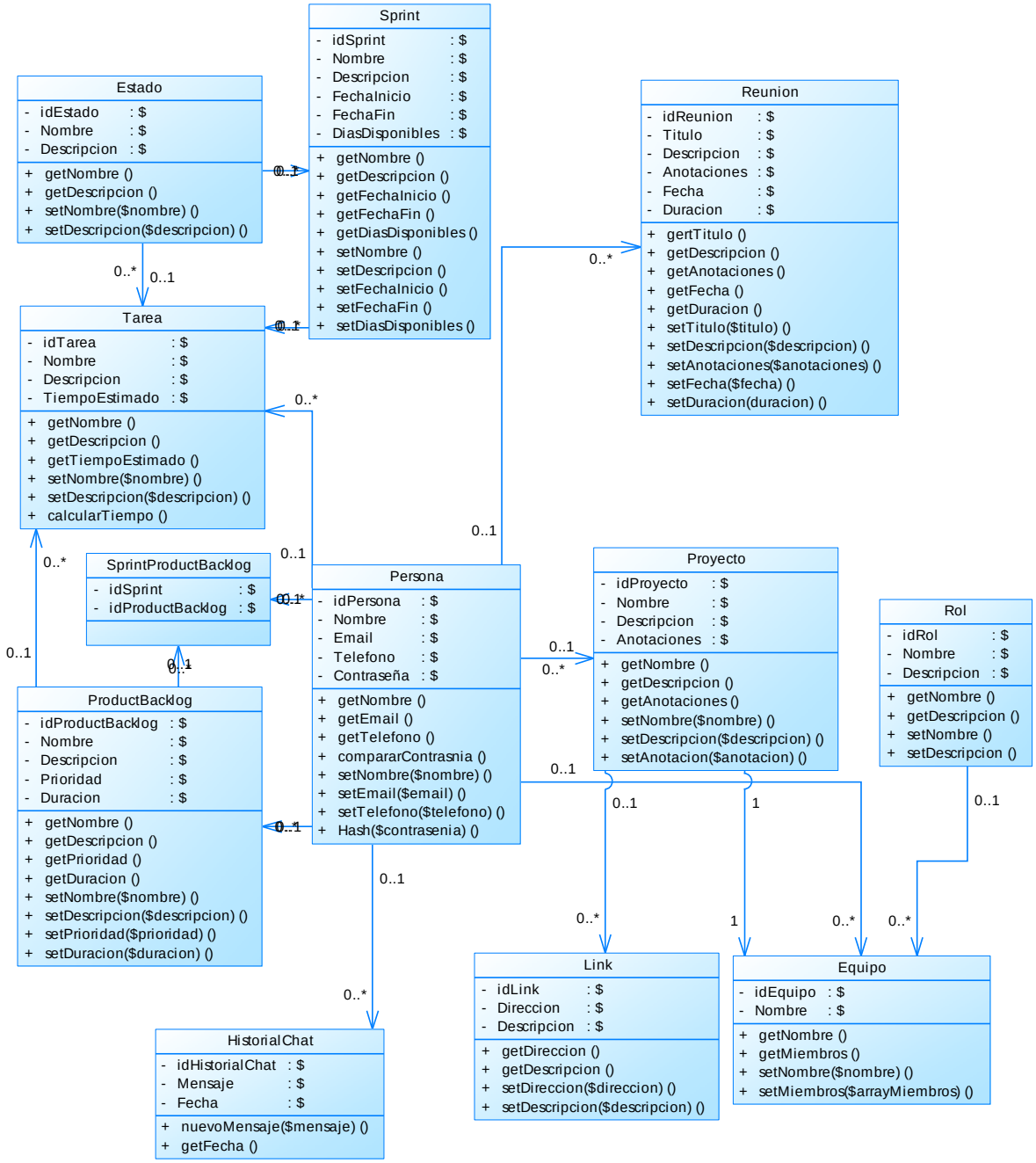
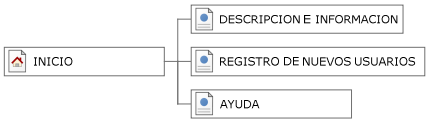


Diagrama de clases



Mapa del sitio web

Parte estática



Parte dinámica

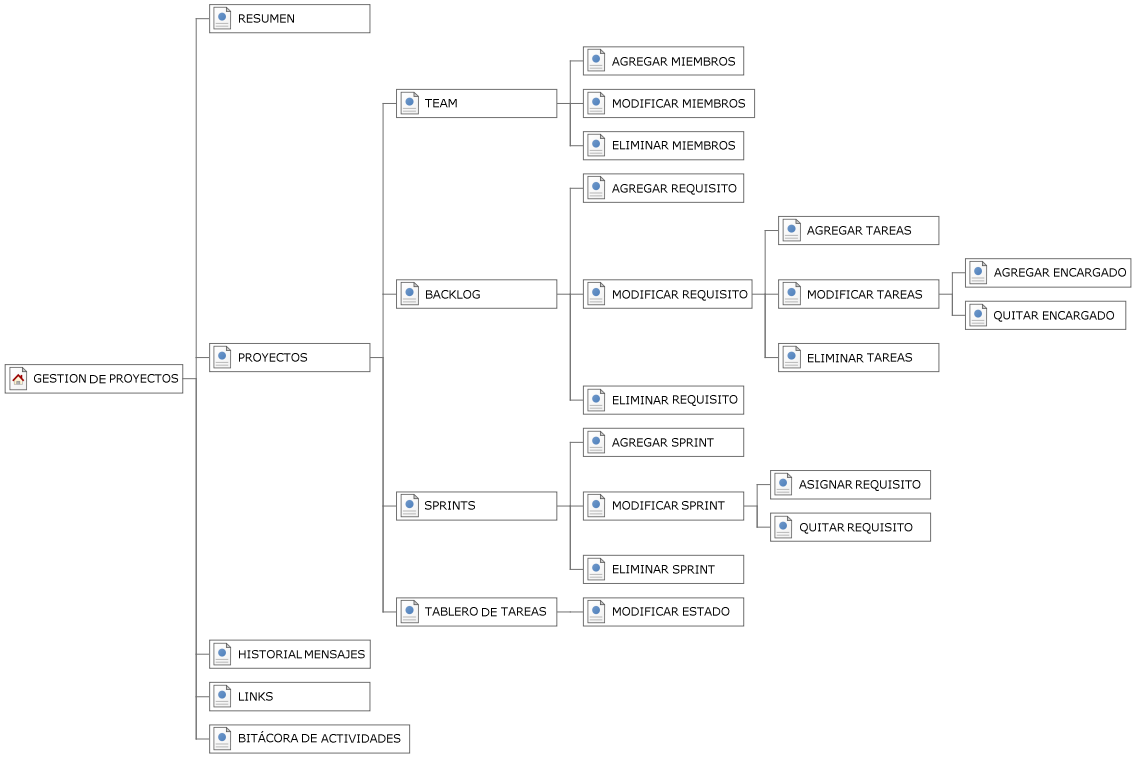
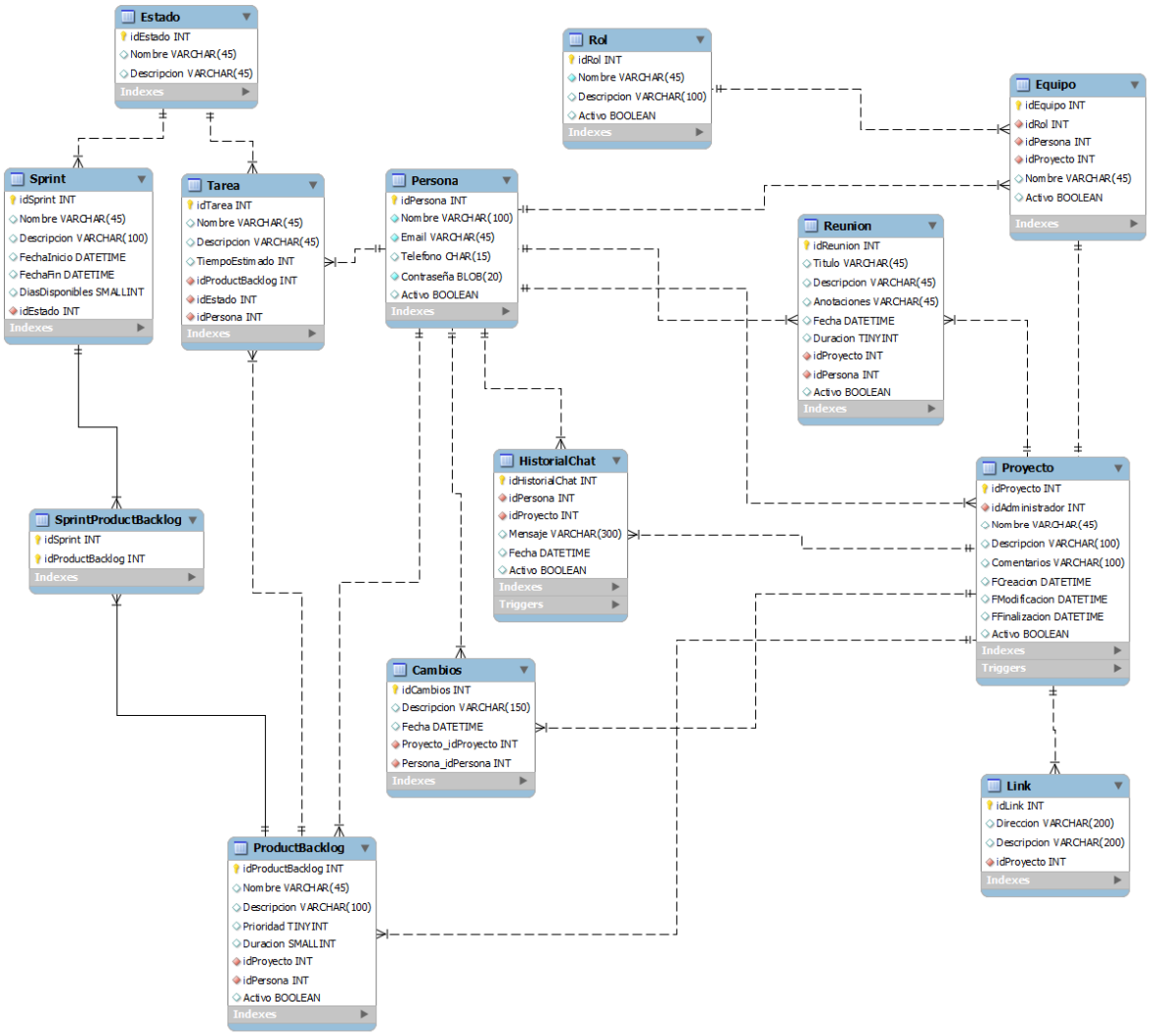


Diagrama de la base de datos



Normalización de código

Historial de modificaciones

Para tener un registro de modificaciones del código fuente se utilizará el servidor de control de versiones que googlecode nos proporciona gratuitamente, aunado a ello se aplicará un plugin para cada cliente (programador).

El lenguaje de programación definido es PHP, con gestor de base de datos MySQL, framework symfony 1.4, IDE netbeans para la edición de archivos PHP y DreamWeaver para la edición de archivos HTML.

Generales

- El software elegido como Entorno de Desarrollo nos proporciona una serie de plugins por default como un corrector ortográfico, indentado y resaltado automático que darán estilo al código.
- El estilo de escritura será lowerCamelCase, salvo los casos de clases Base.
- Las variables siempre deben tratar de declararse al inicio de la clase o función.
- El proyecto será realizado con el paradigma de programación orientado a objetos, dentro del código de la aplicación se contarán con variables, estructuras, clases, componentes, procedimientos, funciones, constantes, etc.
- Se utilizará YAML para los archivos de configuración de symfony. En YAML, la estructura se muestra a través de la sangría, la secuencia de elementos se denotan por un guión, y los pares clave/valor están separados por dos puntos. YAML también tiene una sintaxis abreviada para describir la misma estructura con menos líneas, donde los arrays explícitamente se muestran con [] y los hashes o array asociativos con {}.
    - Hay una cosa importante que necesitas recordar cuando estás editando un archivo YAML: **la indentación debe hacerse con uno o más espacios en blanco, pero nunca con tabulaciones**.
- También se utilizará una arquitectura de programación por capas **MVC**:
    - La capa **Modelo** define la lógica de negocio (la base de datos pertenece a esta capa). Symfony guarda todas las clases y archivos relacionados con el modelo en el directorio lib/model/.
    - La **Vista** es con lo que el usuario interactúa (un motor de plantillas es parte de esta capa). En Symfony, la vista es principalmente la capa de plantillas PHP. Estas son guardadas en varios directorios templates/.
    - El **Controlador** es la pieza de código que llama al **Modelo** para obtener algunos datos que le pasa a la Vista para la presentación al cliente.
- La estructura de directorios que symfony crea se detalla a continuación (en el directorio raíz del proyecto):
    - apps/ Hospeda todas las aplicaciones del proyecto
    - cache/ Los archivos en caché
    - config/ Los archivos de configuración del proyecto

- lib/ Las bibliotecas y clases del proyecto
- log/ Los archivos de registro
- plugins/ Los plugins instalados
- test/ Los archivos de pruebas unitarias y funcionales
- web/ El directorio raíz web

Bajo el directorio `apps/frontend` se crea la siguiente estructura de directorios para el frontend de la aplicación

- config/ Los archivos de configuración de la aplicación
- lib/ Las bibliotecas y clases de la aplicación
- modules/ El código de la aplicación (MVC)
- templates/ La plantilla global

A continuación se procede a detallar cada uno de los componentes a utilizar, y la convención a usar:

Relaciones

En un esquema lógico de datos, son conocidas como entidades, estas representan las tablas donde se almacenarán las tuplas o registros. Está pendiente definir si se utilizarán esquemas. El nombre de la relación debe estar en singular y ser significativo.

Clases

Debido a que se utilizará una arquitectura de programación por capas con el ORM doctrine, el nombre de las clases tomará el siguiente formato que doctrine aplica:

<Nombre>

<Nombre> debe escribirse en formato lowerCamelCase.

P. E. La clase que relaciona proyectos con personas se llamaría ***proyectosPersonas.***

Las clases con nombre `Base<NombreClase>` (`p.e.` ***BaseProyecto*** que es la clase de manejo de proyectos) del directorio `lib/model/base/` son las que se generan directamente a partir del esquema de base de datos por medio del ORM. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.

Por otra parte, las clases de objetos propias que están en el directorio `lib/model` heredan de las clases con nombre `Base`. Estas clases no se modifican cuando se regenera el modelo en caso de cambios, por lo que son las clases en las que se añaden los métodos propios (clases personalizadas).

Funciones

Las funciones deben estar de la siguiente manera

<Nombre>(<Parametros>)

<Nombre> es el indicador de la función, debe estar en formato lowerCamelCase.

<Parámetros> Son los parámetros de la función, deben seguir la convención de declaración de variables.

P. E. la función foo que recibe n parámetros
foo($primerParametro, $segundoParametro, /* ..., */ $ nParametro)

La instrucción return debe ser utilizada una sola vez, y será al final de la subrutina.

Los métodos accesores y modificadores serán identificados con el prefijo ***get*** y ***set*** respectivamente seguido del Nombre del campo.

Constantes

Si se da el caso, para constantes globales y locales se aplicará el siguiente formato:

<NOMBRE>

<NOMBRE> será el identificador de la constante, debe estar totalmente en mayúsculas.

P.E. la constante PI: ***define("PI", 3.1416);***

Variables

Este formato se aplica a las instancias de clases y variables en general:

$<Nombre>

<Nombre> El identificador de la variable, debe ser en formato lowerCamelCase.

P.E. el nombre de un proyecto: ***$nombreProyecto***

Normalización de Interfaz Gráfica de Usuario

- La definición de estilos se hará con hojas de estilo CSS.
- Se utilizará el patrón de diseño decorador para decorar el contenido mostrado en cada página web a través de layouts y plantillas.

Links

- Los links o enlaces deben tener un texto significativo que identifique su función y sean intuitivos para el usuario.

Botones

Botones de Procesamiento de Peticiones:

- El botón que tiene por objeto aplicar un cambio se llamará “Aceptar” navegar a la página adecuada. El que deshace cambios, se denominará “Cancelar”. Y si es muy necesario, un botón para resetear el contenido de los controles del formulario se llamará “Limpiar”

- Si dentro de un formulario se cuenta únicamente con un botón, este deberá estar centrado horizontalmente, si es más de uno, deben estar ubicados en la esquina inferior derecha.

Botones en general

- La altura del botón debe ser la predefinida por el editor.
- El ancho debe ser el que automáticamente define el contenido.
- Si se utilizan imágenes dentro de un botón, estas deben estar centradas en él y quedar al borde (a los cuatro lados) del botón.
- El texto mostrado por los botones debe ser significativo**.**

Etiquetas

- Las etiquetas deben ser llamativas, según la relevancia que tenga dentro del formulario.
- El tamaño debe ser el que automáticamente define el editor.

CheckBox y RadioButton

- Deben estar ubicados dentro de un GroupBox, y cada GroupBox debe tener una categoría de estos controles.

TextBox

- Todos los textbox deben tener alineación vertical centrada.
- Todo contenido numérico debe estar alineado a la derecha.
- Los textBox de valores monetarios deben mostrar el símbolo de la moneda.
- Los textBox para los que sea necesario deberá colocarse un valor por defecto.

Validación de la calidad

Debido a que las pruebas de calidad están presentes en todo el ciclo de desarrollo del software, es necesario establecer puntos de evaluación basándonos en el cronograma ya establecido.

El proyecto se encuentra dividido en los siguientes módulos:

- **Sprint 1** Definición de la arquitectura y modelo de la aplicación web.
- **Sprint 2** Diseño de páginas y flujo de la aplicación.
- **Sprint 3** Módulo de administración de proyectos: En esta parte de la aplicación se gestiona todo lo relacionado con el control general de los proyectos, la creación,modificación y eliminación de estos.
- **Sprint 4** Módulo de administración de personal: Este módulo será el encargado de gestionar a los integrantes del equipo de trabajo, dando la opción de poder crear nuevos perfiles para nuevos miembros, modificar los datos personales de estos y si la situación lo requiere eliminar o dar de baja a cualquier persona del equipo.
- **Sprint 5** Módulo de administración de roles: Gestiona las responsabilidades de los integrantes del equipo de trabajo, asignando puestos, editando el número de integrantes en cada puesto y removiéndolos cuando se desee.Tomando en cuenta los puestos de ScrumMaster, Team, StakeHolders y ProductOwner.
- **Sprint 6** Módulo de administración de documentación:Encargado del control de la documentación de del desarrollo del proyecto, esto incluye al ProductBacklog (pila del product,), SprintBacklog (pila del sprint) y todas las anotaciones e información relevante de las reuniones llevadas a cabo en el proyecto. Llevando el seguimiento de la creacion, edicion y eliminación de todos los tipos de documentación antes mencionados.
- **Sprint 7** Módulo de administración de reuniones:Tendrá a bajo su cargo la organización y calendarización de todas las reuniones que sean pertinentes al proyecto, como la reunión para la planificación del Sprint (Sprint Planning Meeting), la Reunión Diaria (Daily Scrum), la reunión Scrum de Scrum, y las reuniones de revisión y retrospectiva del Sprint. Dando la opción de crear horarios y asignar las fechas y contenidos de cada reunión.
- **Sprint 8** Módulo de comunicación entre miembros del equipo:Control de el chat disponible entre los integrantes del equipo que se encuentren conectados y también se toman en cuenta los links asociados proveídos por los desarrolladores del proyecto.
- **Sprint 9** Mejora de presentación del sitio.

La forma de evaluar la calidad del software será utilizando checkpoints en cada tarea, a continuación se definen:

Número de Sprint	Tareas	Objetivos	Fecha (2012)
1	Creación de casos de uso. Diseño del modelo relacional. Diseño de diagrama de actividades. Diseño de diagrama de clases Exploración intuitiva de Symphony, php, html y javascript	Definir claramente la estructura del software a desarrollar. Conocer las herramientas a implementar durante el desarrollo. Implementación de MVC	28/08 al 03/09
2	Mapa del sitio Diagrama de Estructuras Instalación y configuración de apache, mySQL, symfony, y otras herramientas a utilizar.	Establecer el flujo del sitio web. Establecer claramente las características de cada módulo.	4/09 al 13/09
3	Crear un listado de características del módulo. Aplicar las características. Retroalimentar - cambios Creación del módulo de Administración (usando arquitectura MVC) Pruebas de rendimiento al modelo.	Conocer las especificaciones del módulo de administración. Aumentar calidad durante el desarrollo.	14/07 al 27/09
4	Definición de todas las características del módulo. Manejo de datos de las entidades para su almacenamiento íntegro y seguro. Relación de persona con su respectivo puesto en el proyecto asignado. Análisis e implementación de las características encontradas. Desarrollo de pruebas de las funcionalidades del módulo para	Lograr un control minucioso sobre el personal de trabajo de cada proyecto, incluyendo gestión de entidades y características de estas.	12/10 al 25/10

	verificar el nivel de eficiencia de este.		
5	Listar los requerimientos de trabajo de este módulo. Relacionar de manera íntegra y correcta a la persona o grupo de personas que tomaran la responsabilidad del rol elegido. Implementar opciones de gestión de personal dentro de los roles. Evaluacion de estandares para el funcionamentiento del módulo.	Almacenar y definir correctamente las responsabilidades de cada integrante del equipo de trabajo del proyecto.	26/10 al 08/11
6	Enlistar las características a implementar sobre la gestión de tareas del proyecto (características, tareas, asignación de responsabilidades de desarrollo y planificación). Implementar el log de cambios del proyecto.	Implementar optimización de consultas SQL. Implementar transacciones. Implementar triggers y procedimientos almacenados.	09/11 al 15/11
7	Crear la lista de tareas a implementar para el módulo de reuniones. Implementar las operaciones CRUD (Crear, Obtener, Actualizar y Borrar -del original en inglés **C**reate, **R**ead, **U**pdate and **D**elete-) para el módulo de reuniones.	Conocer la interfaz ORM-Vista. Conocer los objetos de validación de symfony. Conocer el módulo de seguridad de interfaces de symfony. Familiarizarse con las interfaces drag and drop en HTML.	16/11 al 23/11
8	Lista de características para el módulo de comunicación entre miembros del equipo. Aplicar las características. Pruebas de rendimiento.	Conocer las especificaciones del módulo. Permitirles a los usuarios tanto el compartir documentos como la comunicación instantánea mediante el chat.	24/11 al 01/12

9	Con un Benchmarking ofrecer un mejor diseño de la página. Analizar qué características nuevas puede ofrecer la página y cuales se pueden mejorar.	Ofrecer a los usuarios un manejo fácil de las herramientas que proporciona la metodología SCRUM.	01/12 al 08/12